НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.934.5

До захисту допущено
В. о. завідувача кафедри ММСА

О. Л. Тимощук
«___» _____ 2020 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 124 Системний аналіз
на тему: «Рангова система на основі навчання з підкріпленням»

Виконав:
студент ІІ курсу, групи КА-91мп
Сайног Олексій Максимович _____

Керівник:
директор ІПСА, д. ф-м. н.
проф. Касьянов П. О. _____

Рецензент:
професор кафедри інт. та диф. рівнянь
КНУ ім. Тараса Шевченка, д. ф-м. н.,
проф. Капустян О. В. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань

Студент _____

Київ
2020

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – другий (магістерський)
Спеціальність (спеціалізація) – 124 «Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. Л. Тимощук
«___» _____ 2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Сайногу Олексію Максимовичу

1. **Тема дисертації:** «Рангова система на основі навчання з підкріпленням», науковий керівник дисертації Касьянов Павло Олегович, доктор фізико-математичних наук, професор, затверджені наказом по університету від «02» листопада 2020 р. № 3182-с
2. **Термін подання студентом дисертації:** 14 грудня 2020
3. **Об'єкт дослідження:** Навчання з підкріпленням
4. **Предмет дослідження:** Рейтингові системи
5. **Перелік завдань, які потрібно розробити:**
   1) Проаналізувати предметну область та існуючі підходи до рейтингових моделей.
   2) Використовуючи методи навчання з підкріпленням побудувати рейтингові моделі та порівняти їх результати.
   3) Зробити висновки.
   4) На основі отриманих результатів розробити стартап-проект.
6. **Орієнтовний перелік графічного (ілюстративного) матеріалу:**
   1) Рисунки
   2) Таблиці
   3) Блок схеми
7. **Дата видачі завдання:** 1 вересня 2020 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Огляд літератури за тематикою	1 вересня – 1 жовтня	
2	Збір матеріалів по проблемі магістерської дисертації	1 жовтня – 7 жовтня	
3	Вивчення матеріалів по темі магістерської дисертації	7 жовтня – 1 листопада	
4	Аналіз існуючих методів розв’язування задачі	1 листопада – 7 листопада	
5	Написання програмного продукту для розв’язування задачі	7 листопада – 21 листопада	
6	Тестування продукту на існуючих даних	21 листопада – 28 листопада	
7	Оформлення звіту магістерської роботи	28 листопада – 6 грудня	

Студент О. М. Сайног

Науковий керівник дисертації П. О. Касьянов

РЕФЕРАТ

Магістерська дисертація: 93 с., 16 рис., 36 табл., 9 джерел, 2 додатки.

У магістерській дисертації досліджуються методи побудов рейтингових моделей для спортивних подій, що базуються на навчанні з підкріпленням.

Було розглянуто вже існуючі підходи до розробки рейтингових моделей, наведено математичну основу описаних моделей та викладено основні алгоритми, що застосовуються при реалізації.

У роботі також розглянуто метод байєсівської оптимізації, як основний до використання в роботі. Описано багато можливих варіацій реалізації цього методу, а саме різні стратегії обрання наступної точки дослідження та кореляційні функції гаусівських процесів.

Практичне дослідження побудовано на створені рейтингової системи для передбачення результатів тенісних матчів ATP (Асоціація тенісистів професіоналів) турнірів починаючи з 2008 року. Дані було взято з відкритих джерел.

Отримані результати було проаналізовано, оцінено та порівняно. Результати порівняння викладено в висновках.

На основі розглянутої теми розроблено елементи стартап-проекту.

РЕЙТИНГОВІ МОДЕЛІ, РАНГОВІ МОДЕЛІ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, БАЙЄСІВСЬКА ОПТИМІЗАЦІЯ, Q-НАВЧАННЯ, БЕТТІНГ.

ABSTRACT

Master's Thesis: 93 pages, 16 figures, 36 tables, 9 sources, 2 supplements

The master's dissertation investigates the methods of constructing rating models for sporting events based on reinforcement learning.

Existing approaches to the development of rating models were considered, the mathematical basis of the described models is given and the main algorithms used in the implementation are stated.

The paper also considers the method of Bayesian optimization as the main one for use in the work. Many possible variations of the implementation of this method are described, namely different strategies for choosing the next research point and correlation functions of Gaussian processes.

The practical study is based on the creation of a rating system to predict the results of tennis matches ATP (Association of Professional Tennis Players) tournaments since 2008. Data were taken from open sources.

The obtained results were analyzed, evaluated and compared. The results of the comparison are presented in the conclusions.

On the basis of the considered theme elements of the startup project are developed.

RATING MODELS, RANKING MODELS, REINFORCEMENT LEARNING, BAYESIAN OPTIMIZATION, Q-LEARNING, BETTING.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БО – Байєсівська оптимізація

MVP – Minimum viable product

API – Application programming interface

ATP – Association of Tennis Professionals

ВСТУП

Магістерська робота присвячена розробці ігрових рейтингових систем на основі методів навчання з підкріпленням.

Рейтингові системи, що розглядаються в даній роботі є актуальними для ігрової сфери бізнесу, що з огляду на нещодавнє оновлення законодавства стане корисним науковим і практичним надбанням в цій сфері. Розробка власних рейтингових систем забезпечить для букмекерських контор незалежність від зовнішніх експертних оцінок та створить чітке розуміння того, на основі чого прогнозуються ті чи інші ігрові результати.

В роботі реалізовано приклади рейтингових системи на основі навчання з підкріплення. Основним питанням стало те, за яким принципом оновлювати рейтинг після отримання результату змагання. Перші дві моделі є побудованими на байєсівській оптимізації і є відносно швидкими, але менш гнучкими до збільшення кількості параметрів за якими робиться висновок про розмір оновлення рейтингу. Для вирішення проблеми збільшення параметрів було побудовано ще дві моделі, але вже за допомогою Q-навчання. Саме результати цих моделей є найбільш успішними з розглянутих.

Всі розглянуті моделі було побудовано даних ліги ATP (Асоціація тенісистів-професіоналів) починаючи з 2008 року, що було отримано з відкритих джерел.

Враховуючи актуальність та новизну розглянутих моделей виникла необхідність а розробці стартап-проекту на їх основі. Та хоча процес розробки власної системи є тривалим та коштовним, за рахунок серверів під базу даних подій та обчислювальних потужності для роботи рейтингових моделей, її результат є безумовно прибутковим, а головне легко підтримуваним при наявності якісної документації та обізнаних співробітників, що проводитимуть її постійну підтримку.

РОЗДІЛ І. ПОСТАНОВКА ЗАДАЧІ

1.1 Загальне формулювання

Нехай маємо історію деяких двосторонніх* змагань. Історія включає в себе хронологічну інформацію про учасників та результати. Необхідно побудувати модель, що обчислює ймовірність перемоги учасника в змаганні на поточний момент часу.

Загальним підходом до вирішення описаної задачі є присвоєння кожному учаснику рейтингу, що оновлюється після кожного змагання з участю цього учасника. В залежності від результати рейтинг підіймається (учасник переміг) чи опускається (учасник не переміг). Величина підйому чи спуску визначається функцією оновлення, що в свою чергу залежить від помилки передбачення результату змагання.

Для визначення ймовірності перемоги учасника чи команди використовується обрана функція від рейтингів.

1.2 Існуюча література

Розглядаючи ігрові рейтингові моделі слід врахувати їх нішовість та відсутність масовості літературних джерел.

На даний момент всі ігрові рейтингові системи поділяють за наступними рейтинговими методами:

1. Перестановка позицій.

Іноді трапляється, що сильні команди програють слабшим, не завдяки власній безсилості, а певному збігу випадкових обставин. Саме

---

* Двосторонні змагання – ті, в яких учасників поділяють на дві протиборчі сторони.

такі результати вносять розлад в процес оцінки сил противників. Для зменшення їх впливу розглядають додаткові дані по цих змаганнях або взагалі не розглядають цю подію. Спортивний оглядач і письменник Грег Істенбрук навіть створив збірник «якісних» ігор до якого ввійшли лише надійні на його думку перемоги [1].

«Я переглянув перші кілька тижнів ігор і переробив записи всіх, позначаючи кожну гру або законною перемогою/програшем, або надскладною перемогою/програшем, або ‘так-сяк грою’. І якщо ще щось траплялося у цій грі з несподіваними наслідками – перемога у поверненні, провалена перевага, серйозна дисфункція, інше – я теж це позначив»

- Білл Сіммонс, спортивний письменник, Grentland [2]

2. Очікування Піфагора

Застосовує формулу, що оцінює відсоток змагань, які команда «мала» виграти, виходячи з кількості очок, які вона набрала чи пропустила.

$$\text{Частота перемог} = \frac{(\text{набрані очки})^k}{(\text{набрані очки})^k + (\text{пропущені очки})^k}$$

де k – власний коефіцієнт для кожного спорту. Порівнюючи отриманий відсоток зі справжнім визначають поняття «перевиконання» і «недовиконання» результату.

Назва походить зі схожості формули до формули Піфагору, враховуючи те, що її вперше застосували до бейсболу, де $k = 2$. [3]

3. Обмін очками рейтингу

Головною ідеєю, є збільшення рейтингу переможця за рахунок віднімання очок в переможеного. За таким принципом працює оригінальна модель Ело для шахістів [4] та сучасні футбольні рейтинги НФЛ. [5]

Саме до цього типу рейтингових методів належатимуть моделі, що будуть побудовані в наступних розділах.

4. Розв’язування системи рівнянь

Деякі дослідники, такі як Мет Міллс використовують ланцюги Маркова для моделювання футбольних ігор. [6] Також алгоритм Google PageRank адаптований до ранжування футбольних команд. [7]

Головними міркуваннями, які розглядаються в цих рейтингових моделях є:

1. Врахування переваги гри на домашньому полі.
2. Складність суперника відносно інших команд чи учасників при малій кількості зустрічей.
3. Побудова моделі не на результаті, а на різниці очок.
4. **Використання інформації про ігровий процес.**
5. Врахування складу команд та їх заміни.
6. **Проблема холодного старту.**

Товстим шрифтом виділені розглянуті нами проблеми.

1.3 Рейтинг

Різні типи моделей обирають різні поняття рангів (рейтингів):

- Раціональне число (модель Ело)
- Позитивне раціональне число (модель Бредлі-Террі [8])
- Гаусів розподіл (рейтингова система True Skill [9])

В цій роботі за область значень рейтингів оберемо множину раціональних чисел. Як наслідок, функція оновлення рейтингів буде залежати не від значень рейтингів, а від їх різниці.

1.4 Функція підрахунку ймовірностей

Більшість існуючих рейтингових моделей мають схожу функцію підрахунку ймовірності перемоги учасника.

З моделі Бредлі-Террі:

$$\mathbb{P}\{A \text{ переможе } B\} = \frac{R_A}{R_A + R_B}$$

де R_A та R_B – рейтинги гравців A та B відповідно.

В моделі Ело використовується та сама формула, але з заміною:

$$\tilde{R}_i = 10^{\left(\frac{R_i}{400}\right)}$$

Для власної моделі оберемо власну заміну:

$$\tilde{R}_i = e^{\frac{R_i}{S}} \tag{1.1}$$

де S – коефіцієнт нормування рангів, що є гіперпараметром моделі.

А отже маємо наступну функцію підрахунку ймовірності:

$$\mathbb{P}\{A \text{ переможе } B\} = \frac{\tilde{R}_A}{\tilde{R}_A + \tilde{R}_B} = \frac{e^{\frac{R_A}{S}}}{e^{\frac{R_A}{S}} + e^{\frac{R_B}{S}}} = \frac{1}{1 + e^{\frac{R_B - R_A}{S}}},$$

інакше кажучи ми звели загальну формулу до вигляду:

$$\mathbb{P}\{A \text{ переможе } B\} = sigmoid\left(\frac{R_A - R_B}{S}\right).$$

1.5 Функція оновлення

Найголовнішою складовою будь-якої рейтингової системи є принцип, за яким оновлюються рейтинги. Величина оновлення рейтингу повинна залежати від різниці результату та ймовірності. Також слід врахувати кількість оновлень рейтингу: чим більше рейтинг оновлювався, тим більше його можна вважати достовірним.

Для побудови нашої модель візьмемо наступну функцію оновлення:

$$R_A \coloneqq R_A + K \cdot (S_A - \mathbb{P}\{A \text{ переможе } B\}),$$

де:

- $R_A -$ рейтинг гравця A.
- $K -$ коефіцієнт оновлення.
- $S_A -$ результат змагання для гравця A: $0 -$ поразка, $1 -$ перемога, $0.5 -$ нічия.

В практичній частині роботи буде розглянуто різні правила та моделі вибору коефіцієнту оновлення K. Саме вибір цього коефіцієнту в залежності від наявних даних сприяє покращенню роботи моделі. А отже всі подальші дослідження в роботі будуть спрямовані на побудову моделі вибору коефіцієнту оновлення.

Перевагою розглянутих в подальшому моделей вибору коефіцієнта оновлення є використання більшої кількості наявних даних, таких як:

1. Поточний рейтинг учасника.
2. Кількість оновлень рейтингу.
3. Перевага перед іншим учасником (наприклад різниця набраних очок).

Всі ці показники роблять рейтингову модель більш точною та стійкою в часі.

1.6 Загальна схема розробленої моделі

На рисунку 1.1 зображено загальну схему передбачень та оновлення рейтингової системи.

Позначення:

- s – результат змагання між A та B.
- R_A, R_B – рейтинги учасників A та B відповідно.
- $\vec{\beta}$ – додаткова інформація про змагання (перевага переможця, кількість оновлень, тощо).
- F_{onov} – функція визначення коефіцієнту оновлення.

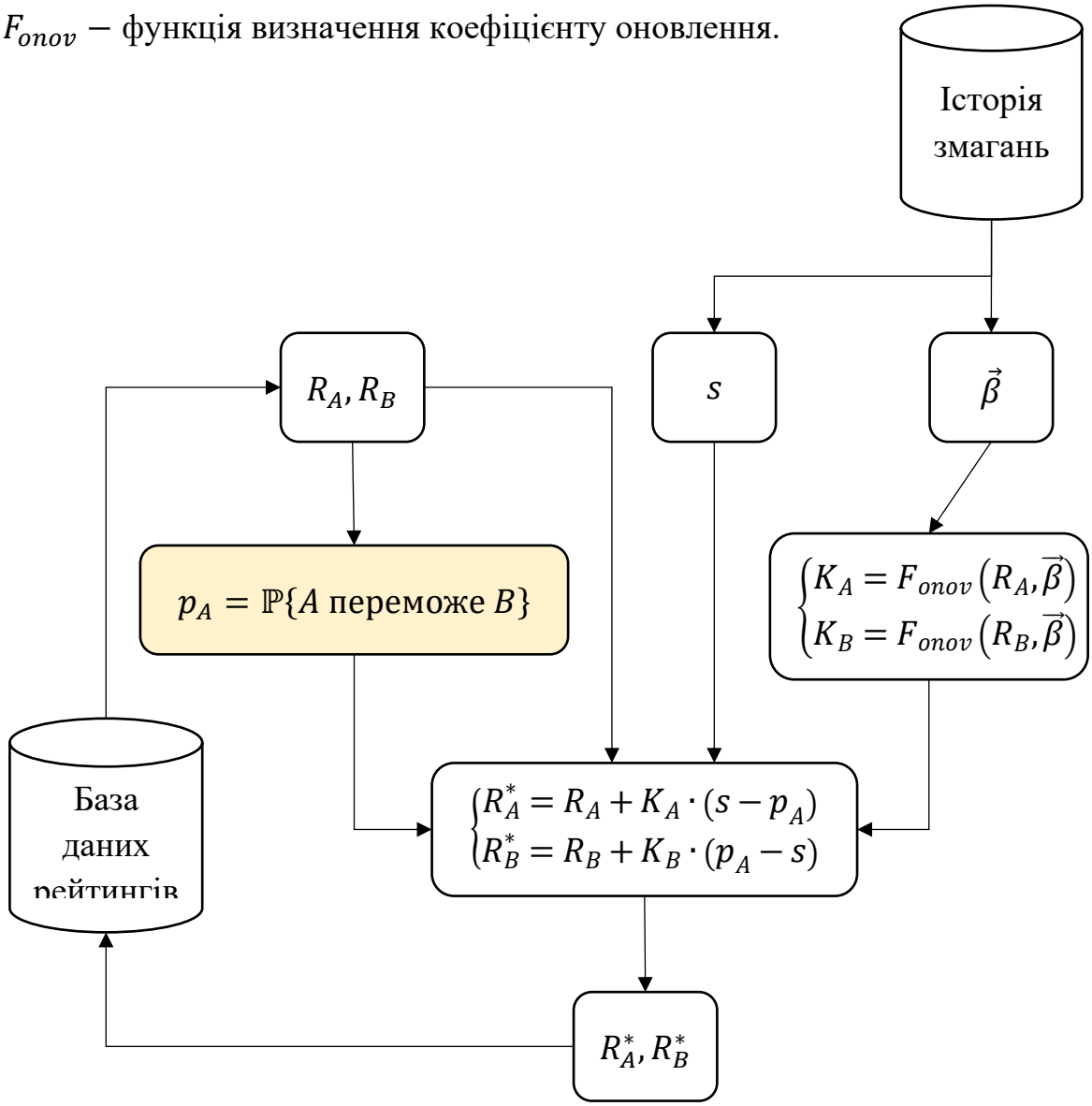

Рисунок 1.1 – Загальна схема розробленої моделі.

1.7 Оцінювання моделі

Оцінювання результатів моделі будемо проводити за двома критеріями:

1. Перехресна ентропія передбачень фіналів матчів на основі рейтингів та фактичних результатів матчів за 2018-2019 рр.:

$$LogLoss(\vec{p}, \vec{r}) = -\frac{1}{n}\sum_{i=1}^{n}(r_i \ln p_i + (1 - r_i)\ln(1 - p_i))$$

2. Співпадіння фактичного розподілу та розподілу на базі рейтингової системи за 2018-2019 рр. та 2020 рік.

Головним принципом даного критерію є: серед усіх матчів з розподілом сил $(p; 1 - p)$, приблизно $p * 100\%$ матчів повинні бути виграними.

Зображати цей критерій будемо у вигляді малюнку з розподілом на базі рейтингової системи та прямої $y = x$ (рисунок 1.2). Найкращим будемо вважати найближчий до прямої розподіл.

Так як, дані за 2020 рік не входять до навчання, то на графіку за цей рік ми спостерігатимемо за стабільністю системи з часом.

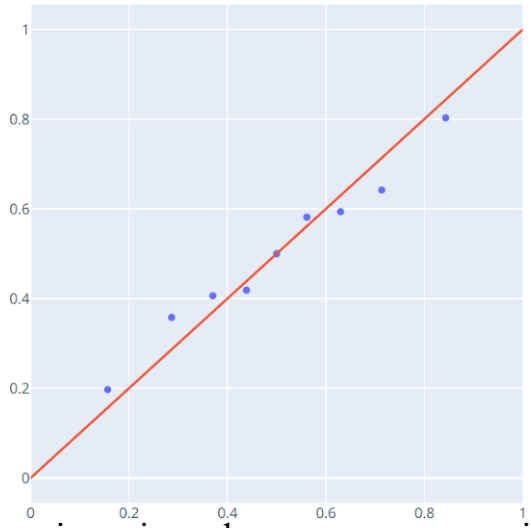

Рисунок 1.2 – Приклад співпадіння фактичного розподілу та розподілу на базі рейтингової системи.

1.8 Висновок до розділу I

В розділі було:

1. Чітко описано проблему, яку пропонується вирішити (підрозділ 1.1).
2. Оглянуту літературу за даною тематикою (підрозділ 1.2).
3. Обрано множину значень рейтингу для рейтингових моделей, що розглядатимуться (підрозділ 1.3).
4. Зафіксовано функції за якими будуть прораховуватися ймовірності (підрозділ 1.4) та оновлюватимуться рейтинги (підрозділ 1.5).
5. Подано загальну архітектуру процесу роботи та тренування (підрозділ 1.6)
6. Описано критерії за якими будуть оцінюватися побудовані моделі (підрозділ 1.7).

РОЗДІЛ ІІ. РЕЙТИНГОВА СИСТЕМА НА ОСНОВІ БАЙЄСІВСЬКОЇ ОПТИМІЗАЦІЇ

2.1 Константна модель вибору коефіцієнту оновлення

Першою розглянемо рейтингову систему, де коефіцієнт оновлення є загальним для будь-якого випадку, а отже вся рейтингова система має 2 параметри, які необхідно оптимізувати.

Але враховуючи той факт, що множення всіх прорахованих рейтингів і нормуючого коефіцієнту з формули 1.1 на одну й ту саму константу не змінює саму рейтингову систему, ми можемо зафіксувати цей гіперпараметр і не проводити оптимізацію по ньому. В усіх розглянутих моделях будемо використовувати коефіцієнт нормування рівний 300.

Оптимізацію проводитимемо за допомогою алгоритму Байєсівської оптимізації.

Отже маємо єдиний параметр – коефіцієнт оновлення. Будемо його шукати на відрізку $[30, 80]$. Процес оптимізації зображено на рисунку 2.1.

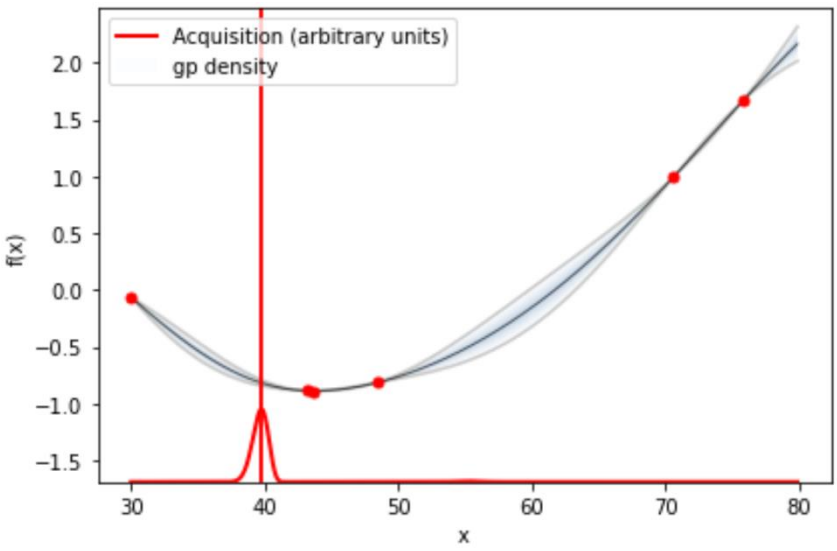

Рисунок 2.1 – БО на відрізку $[30, 80]$. Червоною лінією зображені функція придбання та значення її максимуму.

З рисунку 2.1 видно, що ми можемо звузити відрізок пошуку до $[40, 50]$. Для зручності будемо шукати коефіцієнт з точністю до 0.1, тобто округлюючи значення до першого знаку після к.

Процес пошуку та збіжність алгоритму оптимізації зображено на рисунках 2.2 та 2.3.

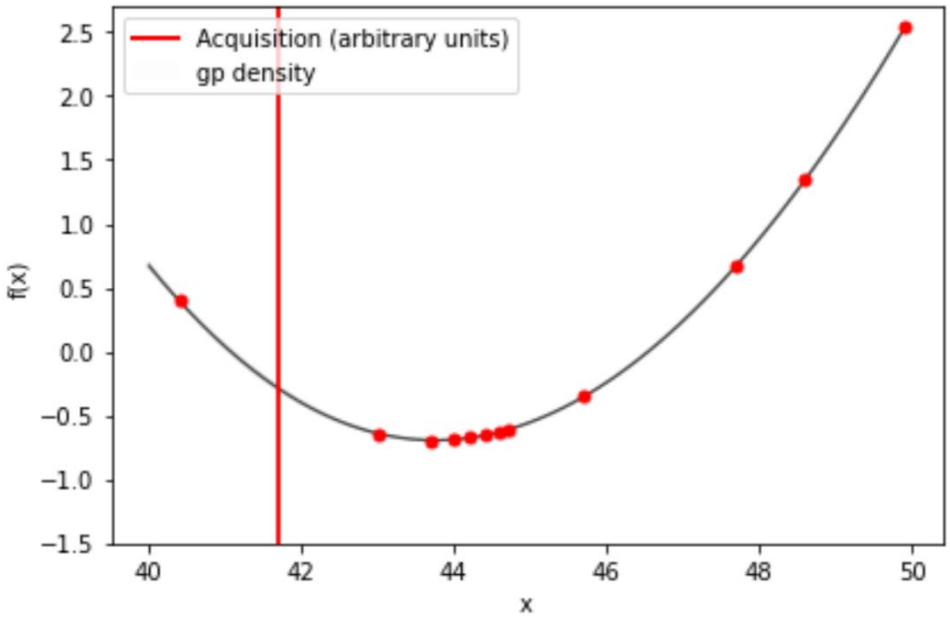

Рисунок 2.2 – БО на відрізку $[40, 50]$.

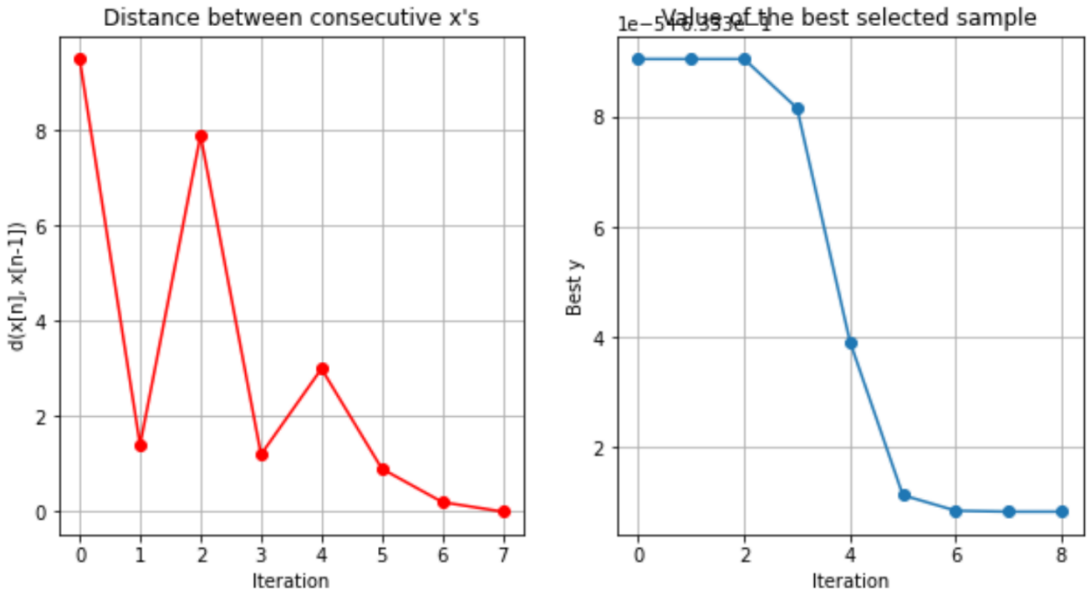

Рисунок 2.3 – Графіки збіжності БО. Перший графік відображає відстані між розглянутими значеннями, а другий – значення функції помилки.

Результат:

– Оптимальним значенням коефіцієнту оновлення: $\boldsymbol{K = 43.9}$.

– Значення перехресної ентропії: **0.633308**.

– Співпадіння фактичного розподілу та розподілу на базі рейтингової системи зображено на рисунку 2.4 та таблицях 2.1 і 2.2.

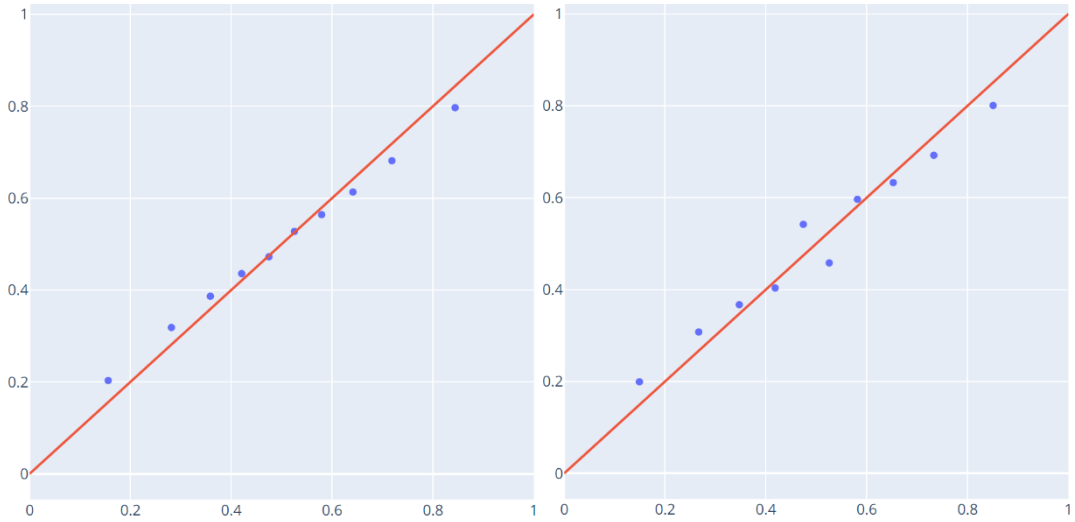

Рисунок 2.4 – Графіки співпадіння фактичного розподілу та розподілу на базі рейтингової системи для 2018-2019 рр. та 2020 року відповідно.

Таблиця 2.1 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2018-2019 рр.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.235]	0.156	0.203
(0.235, 0.323]	0.281	0.319
(0.323, 0.392]	0.359	0.387
(0.392, 0.449]	0.421	0.436
(0.449, 0.5]	0.475	0.472
(0.5, 0.551]	0.525	0.528
(0.551, 0.608]	0.579	0.564
(0.608, 0.677]	0.641	0.613
(0.677, 0.765]	0.719	0.681
(0.765, 1.0]	0.844	0.797

Розмір переваги*	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0087	0.0326	0.0471

* Під розміром переваги мається на увазі модуль різниці ймовірностей для учасників.

Таблиця 2.2 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2020 рік.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.226]	0.149	0.199
(0.226, 0.307]	0.267	0.308
(0.307, 0.386]	0.347	0.367
(0.386, 0.449]	0.418	0.404
(0.449, 0.5]	0.474	0.542
(0.5, 0.551]	0.526	0.458
(0.551, 0.614]	0.582	0.596
(0.614, 0.693]	0.653	0.633
(0.693, 0.774]	0.733	0.692
(0.774, 1.0]	0.851	0.801

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0413	0.0303	0.05

2.2 Модель зі змінним коефіцієнтом оновлення в залежності від поточного рейтингу та кількості оновлень

Наступним кроком стане розгляд моделі, в якій коефіцієнт оновлення обиратиметься за алгоритмом зображеним на рисунку 2.5.

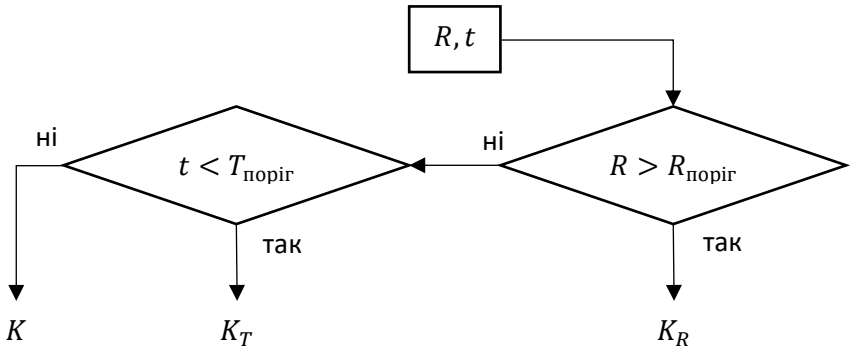

Рисунок 2.5 – Алгоритм вибору коефіцієнту оновлення (K, K_T, K_R) в залежності від поточного рейтингу R та кількості оновлень t. $R_{\text{поріг}}, T_{\text{поріг}}$ – відповідні порогові значення.

Ідея полягає в двох роздумах:

1. Рейтингові учасники мають відмінну поведінку від загальної маси всіх учасників, тому слід врахувати можливість окремого коефіцієнту оновлення для цієї категорії.
2. При малій кількості оновлень рейтинг є менш достовірним, а отже потребує більшого оновлення.

Отже маємо наступний список параметрів до оптимізації:

- $R_{\text{поріг}}$ – порогове значення визначення рейтингового гравця.
  Відрізок оптимізації: $[600, 1000]$, крок 50.
- $T_{\text{поріг}}$ – порогове значення для кількості оновлень.
  Відрізок оптимізації: $[50, 100]$, крок 2.
- K, K_T, K_R – відповідні коефіцієнти оновлення.
  Відрізки оптимізації: $[50, 100], [50, 100], [50, 100]$ з кроком 2.

Процес збіжності БО зображено на рисунку 2.6.

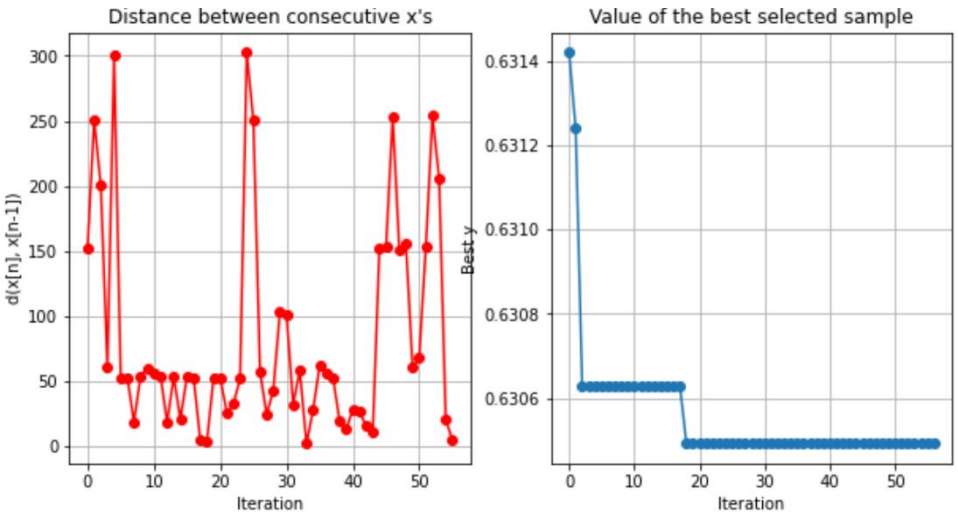

Рисунок 2.6 – Графіки збіжності БО. Перший графік відображає відстані між розглянутими значеннями, а другий – значення функції помилки.

Оптимальні значення параметрів:

$$R_{\text{поріг}} = 750, \qquad T_{\text{поріг}} = 90, \qquad K = 30, \qquad K_T = 64, \qquad K_R = 76.$$

Результат:

– Значення перехресної ентропії: **0.630491**.

– Співпадіння фактичного розподілу та розподілу на базі рейтингової системи зображено на рисунку 2.7 та таблицях 2.3 і 2.4.

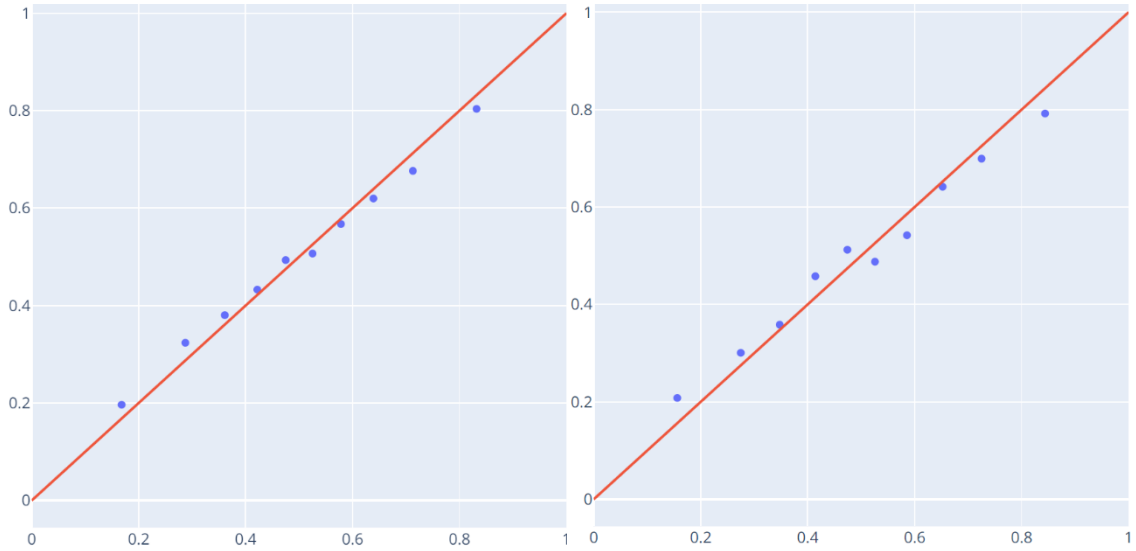

Рисунок 2.7 – Графіки співпадіння фактичного розподілу та розподілу на базі рейтингової системи для 2018-2019 рр. та 2020 року відповідно.

Таблиця 2.3 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2018-2019 рр.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.244]	0.168	0.196
(0.244, 0.328]	0.287	0.324
(0.328, 0.394]	0.361	0.380
(0.394, 0.45]	0.422	0.433
(0.45, 0.5]	0.475	0.493
(0.5, 0.55]	0.525	0.506
(0.55, 0.606]	0.578	0.567
(0.606, 0.672]	0.639	0.620
(0.672, 0.756]	0.713	0.676
(0.756, 1.0]	0.832	0.804

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0147	0.0278	0.0283

Таблиця 2.4 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2020 рік.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.235]	0.156	0.208
(0.235, 0.312]	0.275	0.301
(0.312, 0.383]	0.348	0.358
(0.383, 0.448]	0.414	0.458
(0.448, 0.5]	0.474	0.512
(0.5, 0.552]	0.526	0.488
(0.552, 0.617]	0.586	0.542
(0.617, 0.688]	0.652	0.642
(0.688, 0.765]	0.725	0.699
(0.765, 1.0]	0.844	0.792

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0408	0.0182	0.0518

2.3 Висновок до розділу II

В розділі було побудовано моделі на основі байєсівської оптимізації. Для кожної з моделей було описано процес їх побудови, процес навчання та подано графіки. Результати було відображено згідно обраних у пункті 1.6 критеріїв.

Було викладено причини додавання більшої кількості параметрів до константної моделі. З результатів другої моделі можемо зробити висновок, що врахування більшої кількості наявних даних покращує модель, але й ускладнює. Так для знаходження оптимуму в константній моделі нам знадобилося 8 ітерація, а для покращеної – більше ніж 50.

Слід зазначити, що додані нами умови вибору коефіцієнту оновлення є точковими. Більш цікавим є знаходження оптимального коефіцієнту оновлення від більшої кількості проміжків вхідних параметрів, що розглянемо в наступному розділі.

РОЗДІЛ ІІІ. РЕЙТИНГОВА СИСТЕМА НА ОСНОВІ Q-НАВЧАННЯ

3.1 Q-матриця як модель обрання коефіцієнту оновлення.

Для збільшення ситуацій для яких підбиратимемо власний коефіцієнт оновлення застосуємо алгоритм Q-навчання. Як відомо алгоритм Q-навчання складається з:

- Множина станів (у нашому випадку – множина унікальних проміжків кількості оновлень та поточних рейтингів).
- Множина дій (у нашому випадку – множина коефіцієнтів оновлення)
- Нагороди (у нашому випадку – від'ємне значення перехресної ентропії на кожному кроці, та $1 - LogLoss(\vec{r}, \vec{p})$ для наборів, що не погіршують загальну перехресну ентропію).

Для практичного прикладу розглянемо наступні проміжки:

- Для рейтингів: $(-\infty; 0]$, $(0, 400]$, $(400, 740]$, $(740, +\infty)$.
- Для кількості оновлень: $[0, 89]$, $(89, 300]$, $(300; +\infty)$.

За множину дій візьмемо: $[20, 29, 30, 38, 50, 63, 77, 90]$.

В результаті маємо матрицю $Q \in Matr(12, 8)$, що задається нульовими значеннями на початку. Оптимальні значення представлено в таблиці 3.1.

Таблиця 3.1 – Оптимальні значення коефіцієнту оновлення для проміжків рейтингів (по горизонталі) та кількості оновлень (по вертикалі).

Проміжки даних	$(-\infty; 0]$	$(0, 400]$	$(400, 740]$	$(740, +\infty)$
$[0, 89]$	77	77	29	63
$(89, 300]$	63	20	38	38
$(300; +\infty)$	77	29	20	63

Результат:

– Значення перехресної ентропії: **0.629099**.

– Співпадіння фактичного розподілу та розподілу на базі рейтингової системи зображено на рисунку 3.1 та таблицях 3.2 і 3.3.

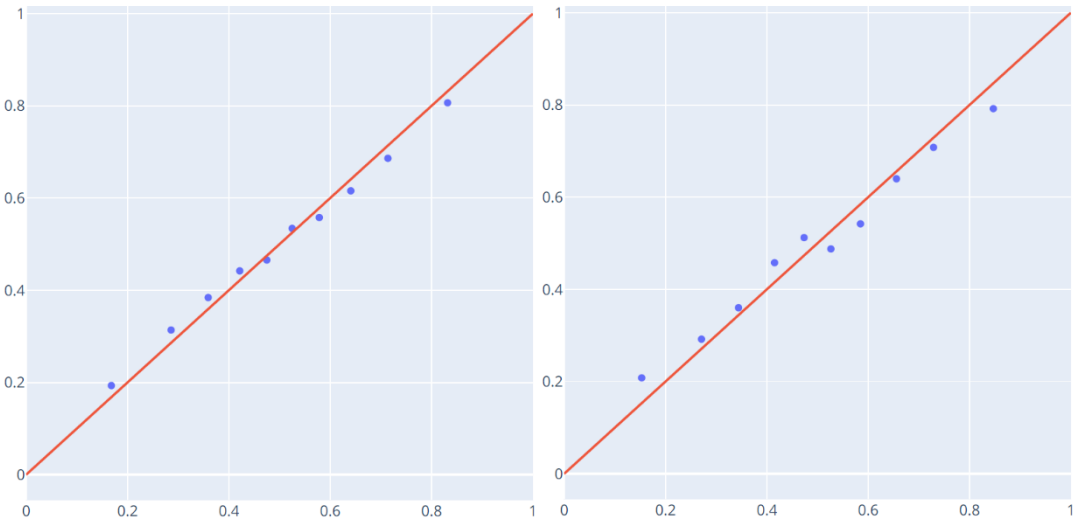

Рисунок 3.1 – Графіки співпадіння фактичного розподілу та розподілу на базі рейтингової системи для 2018-2019 рр. та 2020 року відповідно.

Таблиця 3.2 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2018-2019 рр.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.243]	0.168	0.193
(0.243, 0.324]	0.286	0.314
(0.324, 0.394]	0.359	0.384
(0.394, 0.448]	0.421	0.442
(0.45, 0.5]	0.475	0.466
(0.5, 0.552]	0.525	0.534
(0.552, 0.606]	0.579	0.558
(0.606, 0.676]	0.641	0.616
(0.676, 0.757]	0.714	0.686
(0.757, 1.0]	0.832	0.807

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0149	0.0264	0.025

Таблиця 3.3 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2020 рік.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.235]	0.156	0.208
(0.235, 0.312]	0.275	0.301
(0.312, 0.383]	0.348	0.358
(0.383, 0.448]	0.414	0.458
(0.448, 0.5]	0.474	0.512
(0.5, 0.552]	0.526	0.488
(0.552, 0.617]	0.586	0.542
(0.617, 0.688]	0.652	0.642
(0.688, 0.765]	0.725	0.699
(0.765, 1.0]	0.844	0.792

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0406	0.0184	0.0549

3.2 Використання додаткових даних

В даних історії матчів ми маємо кількість набраних очок кожним гравцем. Порахувавши їх різницю, ми отримаємо деяку оцінку переваги сильнішого гравця над слабшим. Ідея полягає в тому, що якщо учасник перемагає з дуже великою перевагою то відстань між рангами учасників повинна бути значно більшою, ніж у випадку малої переваги.

Отже ми використаємо той самий алгоритм з тими самими відрізками, але додавши стани з наступними значеннями переваги: $(-\infty; -2], (-2, -1], (-1, 1], (1, 2], (2, +\infty)$.

Важливо відмітити, що при оновленні значень матриці використовується максимальне значення для наступного стану. Так, як ми теоретично не маємо наявної інформації про значення переваги в наступному стані, то слід шукати максимум серед усіх можливих значень переваги.

Результат:

– Значення перехресної ентропії: **0.627954**.

– Співпадіння фактичного розподілу та розподілу на базі рейтингової системи зображено на рисунку 3.2 та таблицях 3.4 і 3.5.

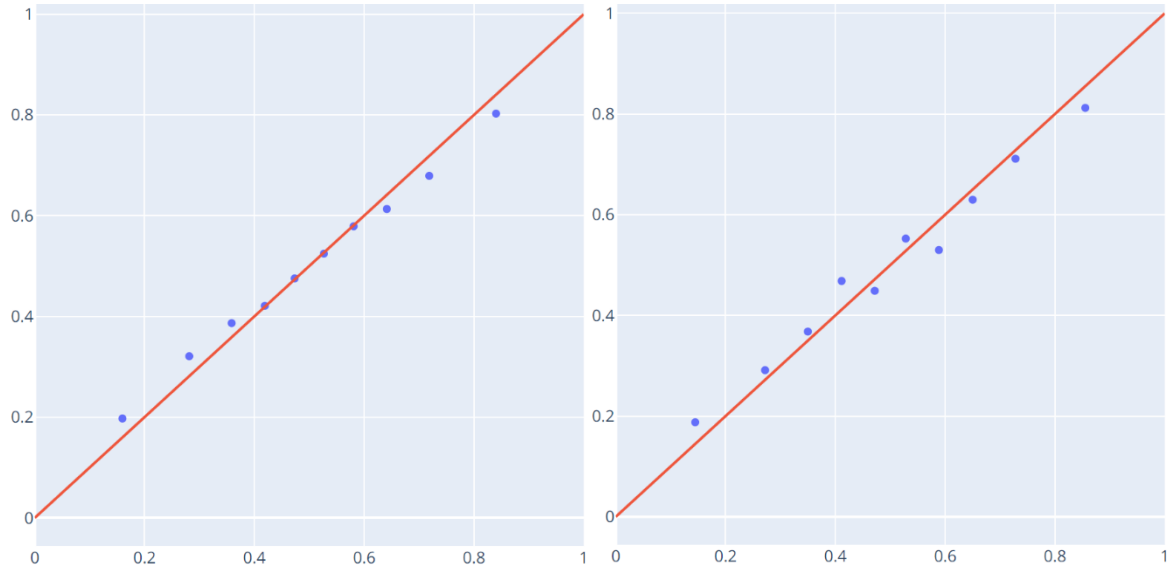

Рисунок 3.2 – Графіки співпадіння фактичного розподілу та розподілу на базі рейтингової системи для 2018-2019 рр. та 2020 року відповідно.

Таблиця 3.4 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2018-2019 рр.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.236]	0.160	0.197
(0.236, 0.323]	0.281	0.321
(0.323, 0.389]	0.359	0.387
(0.389, 0.447]	0.419	0.421
(0.447, 0.5]	0.473	0.475
(0.5, 0.553]	0.527	0.525
(0.553, 0.611]	0.581	0.579
(0.611, 0.677]	0.641	0.613
(0.677, 0.764]	0.719	0.679
(0.764, 1.0]	0.840	0.803

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.002	0.0337	0.0376

Таблиця 3.5 – Дані про відрізки ймовірності, середні значення ймовірності в цих відрізках та середні відсотки перемог за 2020 рік.

Відрізок ймовірностей	Середня ймовірність	Середній відсоток перемог
[0.0, 0.223]	0.145	0.188
(0.223, 0.314]	0.272	0.291
(0.314, 0.381]	0.350	0.368
(0.381, 0.443]	0.411	0.468
(0.443, 0.5]	0.472	0.449
(0.5, 0.557]	0.528	0.553
(0.557, 0.619]	0.589	0.530
(0.619, 0.686]	0.650	0.630
(0.686, 0.777]	0.728	0.711
(0.777, 1.0]	0.855	0.812

Розмір переваги	від 0 до 0.2	від 0.2 до 0.5	від 0.5 до 1
Помилка (MAE)	0.0406	0.0186	0.043

3.3 Висновок до розділу III

В розділі було розглянуто рейтингові системи з моделями вибору коефіцієнту оновлення в залежності від наявних даних про результат чи обставини змагання. Процес навчання було побудовано на алгоритмі Q-навчання.

З результатів побудованих моделей можемо зробити висновок, що додавання додаткових наявних даних робить модель більш точною, але й складнішою до використання. Зі збільшенням кількості врахованих параметрів зростає й розмір Q-матриці, що в свою чергу тягне збільшення ітерацій для знаходження оптимального розв’язку.

РОЗДІЛ 4. СТАРТАП-ПРОЕКТ

4.1 Генерація ідеї

Так як розглянуті нами моделі рейтингових систем мають за мету апроксимацію ймовірності перемог, то найбільш прикладною сферою для них є спортивний чи кіберспортивний беттінг.

Спортивний беттінг — діяльність пов'язана з передбаченням подій та спрямована на отримання прибутку. Гравець самостійно обирає сферу беттінгу та оперує своїми знаннями, вміннями та банком таким чином, щоб мінімізувати вплив випадку.

Тож для стартапу розглянемо ідею створення букмекерської компанії.

Для формування ідеї було проведено наступні кроки:

1. Визначення функції, які повинен виконувати прийнятний варіант виробу.
2. Подання на карті широкого спектра елементарних рішень, тобто альтернативних засобів реалізації кожної функції.
3. Обрання по одному прийнятному елементарному рішенню для кожної функції.

Визначення основних функцій:

1. Розробка програм зі зручним доступом та зрозумілим функціоналом.
2. Підбір лінії з найбільш актуальних спортивних подій.
3. Створення додаткових маркетів для заохочення користувачів.

Визначимо технічних засобів забезпечення основних функцій:

1. Розробка рейтингових систем під кожний вид спорту.
2. Автоматизована чи напівавтоматизована система перевірки документів користувачів.
3. Розробка індивідуальних мікросервісів для обрахунку ймовірностей подій в кожному виді спорту.
4. Оренда та обслуговування обчислювальних потужностей та об’ємів пам’яті.

5. Розробка візуальної та логічної складових загальної програми.

Обмеження:

1. Виконання вимог діючого законодавства щодо вікового цензу користувачів.
2. Безкоштовність використання додатку для користувачів.

Ідея та опис стартап-проекту подано в таблицях 4.1-4.24.

Таблиця 4.1 – Морфологічна карта.

Параметри	Проміжне рішення		
	1-ше	2-ге	3-тє
Рейтингова система	Запозичити в конкурентів	Розробка власної примітивної системи	Розробка власної системи з урахуванням всієї наявної інформації
Система перевірки документів	Ручна перевірка документів	Автоматична перевірка документів	Змішана перевірка з використанням автоматизації як рекомендацій
Мікросервіси обчислення ймовірностей	Один загальний	Для кожного виду спорту	Для кожної ігрової дисципліни
Обчислювальні потужності	Власні комп’ютери	Використання власних серверів	Оренда серверів в хмарі
Програма	Додаток	Сайт	Сайт + Додаток

Отже маємо наступне формулювання стартапу:

1. Букмекерська компанія, що розміщує свої дані та обчислення на хмарних серверах.
2. Користувачі отримуватимуть дані у вигляду сайту чи додатку на їх власний розсуд.
3. Для кожної спортивної дисципліни розроблено мікросервіси, що обчислюють ймовірності подій та повертають кінцеві дані до основної програми.
4. Основою переважної більшості мікросервісів будуть ігрові рейтингові моделі.
5. Для виведення коштів з рахунків користувачу необхідно пройти перевірку даних.

Таблиця 4.2 – Синхронізація завдань.

Етапи	Продукти		
Минуле	Контора як окреме приміщення	Букмекери на стадіонах чи рингах	Ставки на іподромах через касу
Сьогодні	Контора як сайт чи додаток	Онлайн букмекери	Онлайн додатки
Завтра	Моделі на основі машинного навчання та навчання з підкріпленням	Інтернет платформа для букмекерів	Онлайн помічники, що підбирають найкращі ставки
Післязавтра	Моделі на основі штучного інтелекту	Ставки в власній крипто валюті	Штучний інтелект як професійний гравець

4.2 Команда стартапу

Таблиця 4.3 – Оцінювання важливості фактору і внеску кожного учасника.

Фактор	Вага	Технічний директор	Розробник беку	Розробник фронту	Науковець з даних	Трейдер	Маркетолог
Ідея	4	10			7	1	
Розробка бізнес плану	5	8					7
Участь у розробці	7	2	9	8	6	2	
Залученість і ризики	8	2	6	3	6	8	4
Обов’язки	6	10	8	7	7	7	5

Таблиця 4.4 – Визначення дольової участі у стартап проекті кожного учасника.

Фактор	Технічний директор	Розробник беку	Розробник фронту	Науковець з даних	Трейдер	Маркетолог
Ідея	40			28	4	
Розробка бізнес плану	40					35
Участь у розробці	14	63	56	42	14	
Залученість і ризики	16	48	24	48	64	32
Обов’язки	60	48	42	42	42	30
Разом	170	159	112	160	126	97
Частка	20,63%	19,3%	13,59%	19,42%	15,29%	11,77%

Таблиця 4.5 – Обов’язки.

Технічний директор	Розробник беку	Розробник фронту	Науковець з даних	Трейдер	Маркетолог
Розробка загального напрямку розвитку компанії	Розробка логічної частини програмного продукту.	Розробка візуальної частини продукту.	Розробка мікросервісів з обрахунку ймовірностей	Ведення розрахунку подій та введення даних	Розробка стратегій з залучення нових клієнтів та утримання наявних

4.3 MVP

Головна проблема: з розвитком теорії ймовірності, машинного навчання та обчислювальних потужностей користувачі навчилися робити ставки, що приносять втрати для букмекерських компаній.

1-й MVP. Спір на гроші, що основувався лише на власному передчутті. Проблему вирішено, бо досвідчені букмекери мають значний досвід.

2-й MVP. Використання звичайної ймовірності, основуючись на статистиці. Проблему вирішено, бо статистично букмекер залишатиметься в плюсі.

3-й MVP. Використання умовних ймовірностей, тобто врахування наявної додаткової інформації. Проблему вирішено оскільки такий підхід дозволяє отримувати прибутки на більш коротких дистанціях.

4-й MVP. Побудова дерев рішень. Проблему вирішено, так як дерева рішень є більш удосконаленим способом підрахунку умовних ймовірностей.

5-й MVP. Використання нейромереж. Є вирішенням проблеми та більш надійнішим способом ніж використання дерев рішень, так як може гарантувати неперервність апроксимованих значень, а при корегуванні навчання й монотонність.

6-й MVP. Використання штучного інтелекту. Є найкращим рішенням, бо буде можливість врахування всієї наявної інформації, що перевершить можливості людини чи алгоритму.

4.4 Бізнес модель

Таблиця 4.6 – Бізнес-модель «Canvas».

<table>
<tr><th>Ключові партнери</th><th>Ключові види діяльності</th><th>Ціннісна пропозиція</th><th>Взаємовідносини з клієнтами</th><th>Споживчі сегменти</th></tr>
<tr><td rowspan="3">Покупці даних нашої компанії</td><td>Беттінг</td><td rowspan="3">Розширена лінія, дизайн, нижча від конкурентів маржа</td><td>Персональна підтримка, автоматизована перевірка та обслуговування</td><td rowspan="3">Нішовий ринок</td></tr>
<tr><td>Ключові ресурси</td><td>Канали збуту</td></tr>
<tr><td>Інтелектуальні (програма), матеріальні (хмарні сервери)</td><td>Сайт, додаток, конкуренти</td></tr>
<tr><td colspan="3">Структура витрат</td><td colspan="2">Потоки доходів</td></tr>
<tr><td colspan="3">Фіксовані витрати (розробку та оренда),
Нефіксовані (втрата доходів)</td><td colspan="2">Ставки користувачів, доходи від продажу даних</td></tr>
</table>

Таблиця 4.7 – Бізнес-модель «Lean Canvas».

<table>
<tr><td>Проблема</td><td>Рішення</td><td>Унікальна торгова пропозиція</td><td>Прихована перевага</td><td>Споживчі сегменти</td></tr>
<tr><td rowspan="3">Розробка прибут-ких моделей</td><td>Використання методів машинного навчання</td><td rowspan="3">Розширена лінія, нижча від конкурентів маржа</td><td>Покупка обчислених ймовірностей, зручний дизайн</td><td rowspan="3">Нішовий ринок</td></tr>
<tr><td>Ключові метрики</td><td>Канали збуту</td></tr>
<tr><td>Інтелектуальні (ПЗ), матеріальні (хмарні сервери)</td><td>Сайт, додаток, конкуренти</td></tr>
<tr><td colspan="3">Структура витрат</td><td colspan="2">Потоки доходів</td></tr>
<tr><td colspan="3">Фіксовані витрати (розробку та оренда), Нефіксовані (втрата доходів)</td><td colspan="2">Ставки користувачів, доходи від продажу даних</td></tr>
</table>

4.5 Аналіз ринкових можливостей

Таблиця 4.8 – Характеристика потенційних клієнтів.

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних груп клієнтів	Вимоги споживачів
Бажання виграти ставку з найбільшими коефіцієнтами	Клієнти, що бажають поставити ставку	В залежності від досвіду та об’єму коштів	Високі коефіцієнти, зручний дизайн
Потреба у вже обчислених ймовірностях	Конкуренти	В залежності від наявної аудиторії та доходів	Якість та надійність

Таблиця 4.9 – Ринкові можливості та загрози.

Параметри оцінки	Можливості	Загрози
Конкуренція	Постійне вдосконалення власної продукції	Зниження маржі у конкурентів
Економічні фактори	Створення нових допоміжних продуктів	Зниження доходів потенційних клієнтів
Розвиток технологій	Постійне вдосконалення методів, які використовують	Створений новий набагато зручніший метод вирішення проблеми

Таблиця 4.10 – Ступеневий аналіз конкуренції на ринку.

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Чиста конкуренція	Не існує явних лідерів	Спроба виділити переваги даного методу
Світовий рівень конкурентної боротьби	Конкуренти з різних країн світу	Здобути першість в Україні та Європі
Внутрішньогалузева	Конкуренція спостерігається в галузі	Запропонувати новий метод
Товарно-родова	Конкуренція між методами одного роду	Розробка зручних та привабливих функцій
Ціновий характер	Висока маржинальна ставка	Зменшення маржинальної ставки
Не марочна	Для споживачів не має значення бренд	Головним є якісна підтримка основних функцій

Таблиця 4.11 – Аналіз конкуренції в галузі за М. Портером.

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари - замінники
Складові	Наявні букмекер-ські контори	Затрати, досвід, клієнтська база	Відсутні	Кількість ставок, зацікавленість	Маржа
Висновки	Все одно надають можливості автоматизу-вати процес	Вихід на ринок за рахунок масової реклами, стартових бонусів та низької маржі	Є можливість стати постачальни-ком для інших	Клієнти диктують умови: зручність, низька залежність від лімітів	Обмеження встановлен ня високої маржиналь -ної ставки

Таблиця 4.12 – Обґрунтування факторів конкурентоспроможності.

№	Фактори	Обґрунтування
1	Потреба споживачів	Широкий вибір маркетів допоможе знайти найкращі варіанти ставок
2	Простота	Продукт повинні мати зручний дизайн
3	Ціна	Не завищена, а конкурентна маржинальна ставка

Таблиця 4.13 SWOT-аналіз.

Сильні сторони	Слабкі сторони	Можливості	Загрози
Зручність дизайну, широкий вибір маркетів, низька маржа	Велика конкуренція	Вихід на міжнародний ринок, нові функції та продукти	Активізація конкурентів, недостатність коштів

4.6 Стратегія просування на ринку

Таблиця 4.14 – Визначення базової стратегії розвитку.

Стратегія охоплення ринку	Ключові конкурентоспроможні позиції обраної альтернативи	Базова стратегія розвитку
Нашою стратегією захоплення ринку буде захоплення ринку встановленням низької маржі та наявність інноваційних компонентів.	Конкурентними стратегіями до нашої базової стратегії розвитку є: 1) Стратегія лідерства по витратах 2) Стратегія спеціалізації	Як базову стратегію розвитку ми будемо використовувати стратегію диференціації. Наш проект буде сфокусованим на якості, зручності та інноваціях.

Таблиця 4.15 – Визначення базової стратегії конкурентної поведінки.

Чи є проект новатором на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні, але має новаторські складові	Компанія буде шукати нових клієнтів та переманювати від конкурентів	Так, наявні списки маркетів	Для продажу даних буде використана стратегія лідера. За підстратегію оберемо наступальну, щоб збільшити частку ринку. Будуть постійні техніко-економічні вдосконалення.

Таблиця 4.16 – Визначення ключових переваг концепції потенційного товару.

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або майбутні)
Інші компанії не мають ресурсів для якісного обрахунку ймовірностей	Товар пропонує унікальну можливість покупки вже обчислених даних.	Основними перевагами є якість, безпечність та орієнтованість на клієнта.

Таблиця 4.17 – Опис трьох рівнів моделі товару.

Рівні товару	Сутність та складові
I. Товар за задумом	Дані, що являють собою обчислені ймовірності подій.
II. Товар у реальному виконанні	Властивості/характеристики: 1. Середньостатистичне відхилення не більше 3%. 2. Підтримка API.
	Якість: стандарт безпеки інформації CISCO
	Марка: BookProMaker
III. Товар з підкріпленням	До продажу – не потребує особливих навичок.
	Після продажу – не потребує особливих навичок.

Таблиця 4.18 – Визначення меж встановлення ціни.

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Безкоштовно	15% від прибутку	Від 50 тис. дол./ міс.	Від 8% до 12% прибутку

Таблиця 4.19 – Формування системи збуту.

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти будуть купувати продукт згідно контракту	Гарантія та підтримка продукту	Канал нульового рівня	Збут продукції відбувається шляхом API запитів

Таблиця 4.20 – Концепція маркетингових комунікацій.

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Клієнти дізнаються про нові продукції з додатку чи сайту. Також можливе залучення інших способів реклами.	Інтернет, пошта, мобільний зв’язок	SMM, контент - маркетинг	Представлення товару, його позиціювання з метою залучення клієнтів	“Наша якість – Ваші гроші!”

4.7 Бізнес план проекту

4.7.1 Резюме

– **ActPro**

– Характеристика організації, що звертається за наданням засобів:

- найменування організації: **BookProMaker**;
- організаційно-правова форма: **ПрАТ**;
- форма власності: **приватна**;
- кількість розробників / кількість співробітників: **20**;
- статутний фонд: **25 млн. грн.**;
- оборот за останній рік (для діючих організацій): **0 грн.**;
- контактні дані: **info@doggyhome.com , +380661234567**;
- банківські реквізити (для діючих організацій).

Таблиця 4.21 – Приклад банківських реквізитів.

Отримувач коштів	ПрАТ «БукПроМейкер» м. Київ
Код отримувача	31032378
Банк отримувача	АТ "УНІВЕРСАЛ **БАНК**"
SWIFT	EXBSUAUX
Рахунок отримувача (IBAN)	UAH: UA913223130000026008020020371 (980)
	EUR: UA913223130000026408020020371 (978)
	USD: UA913223130000026208020020371 (840)

– прізвище, ім'я, по батькові, вік і кваліфікація керівника проекту: **Сайног Олексій Максимович, 22, CTO**.

– Команда, що працюватиме над проектом, складається з 20 чоловік: **CTO, 2 Senior IOS Dev, 2 Senior Android Dev, 1 Designer, 2 Product Managers, 2 Senior Backend Dev, 2 Senior Frontend Dev, Marketing Analyst, 3 Traders, 2 Automation QA, 2 Senior Data Scientist**.

– Проект є стартапом і буде повністю розроблений від бізнес-плану до розробки і релізу, а також з подальшою його підтримкою.
– Перевага продукції: **Повний цикл розробки і підтримки продукту**.
– Власні ресурси компанії і її поточний фінансовий стан:

На даний момент компанія знаходиться на стартовому етапі, вже залучено кілька інвесторів, наймані працівники, орендований офіс і закуплене обладнання.

– Довгострокові і короткострокові цілі проекту:
  - **Короткострокові цілі: перший реліз продукту до кінця року, після релізу і залучення партнерів, чистий дохід 100 тис. дол. США за перше півріччя після релізу.**
  - **Довгострокові цілі: за наступні 5 років створення нових проектів, залучення більшої кількості працівників, дохід 2 млн дол. США на рік.**
– Потреба в інвестиціях, напрями їх використання, передбачувані джерела фінансування, як вони будуть повертатися інвесторам:

Проект потребує багато інвестицій. Інвесторами можуть бути як звичайні бізнесмени, що вважають цей стартап прибутковим, так і майбутні партнери. Інвесторам буде повертатися доля від доходу після успішної реалізації проекту.

– Наявність ліцензій, сертифікатів, дозволів і т.д.

Приватне акціонерне товариство створено, всі працівники офіційно наймані, триває етап отримання ліцензії на інтелектуальну власність та розробку стартапу.

– Ключові економічні показники ефективності проекту:
  - **Оборотність обігових коштів.**
  - **Рентабельність обігових коштів.**
  - **Відносне вивільнення обігових коштів.**
  - **Рентабельність інвестицій.**
  - **Строк окупності вкладених інвестицій.**
  - **Можливі ризики і система страховок.**

4.7.2 Опис підприємства

- Професійний досвід в обраній галузі.

Частина членів команди мали досвід з розробкою інших стартапів. Інша частина – кваліфіковані спеціалісти, що здатні якісно виконувати свою роботу і підтримувати розробку проекту на високому рівні.

- Профільні професійні досягнення ваші і членів вашої команди.

Майже вся команда має senior позиції, що означає, що вони мали успішний досвід розробки не один рік. Всі члени команди мають вищу освіту, проходили різноманітні курси, що стосувалися їх професійних навичок, які вони будуть застосовувати в роботі. Кожен член команди має рівень знання англійської не нижче Intermediate, що допоможе в свій час вийти на міжнародний ринок.

4.7.3 Опис продукту

Ідеєю нашого стартапу є букмекерська контора, що продає вже обчислені дані іншим конкурентам. Продукт являтиме собою базу даних з вже обчисленими ймовірностями настання подій. В додатку будуть відображатися лінія з матчів чи подій, на які користувачі зможуть поставити ставки. Найважливішою і найзручнішою особливістю цього продукту є якість і надійність.

Назва: ActPro

Призначення: Надання користувачам вже обчислених даних.

Область застосування:

Інші букмекерські контори з невеликими фінансовими статками.

Торгова марка: BookProMaker.

Особливості дизайну: Дизайн даного продукту буде створюватися в залежності від замовника, але з дотриманням умови максимальної простоти та комфорту.

Відповідність стандартам: Товар відповідає всім прийнятим стандартам.

Стадія: Продукт знаходиться на етапі розробки.

Логотип (рисунок 4.1)**:**

Рисунок 4.1 – Логотип продукту.

Продукція повністю відповідає вимогам законодавства, якщо дотримані всі умови і отримані всі ліцензії. В кожній країні люди тримають собак, тому немає жодних невідповідносте до традицій і звичаїв ринку.

Особливості технології вироблення / продажів:

Процес розрахунку ймовірностей відбувається постійно з врахуванням усіх наявних даних. Розраховані ймовірності будуть збережені до бази даних з якої клієнти зможуть їх отримувати через API запити.

Користувачі надаватимуть перевагу конкретно цьому продукту через використання найкращих та найточніших підходів до розрахунку ймовірностей. Надалі, коли аналоги можуть з'явитися, аналітики будуть шукати нові способи просування продукту на ринку з урахуванням маркетингових та економічних трендів.

Надалі, коли аналоги можуть з'явитися, аналітики будуть шукати нові способи просування продукту на ринку з урахуванням маркетингових та економічних трендів.

Щодо передпродажного обслуговування, то це є безпосередньо розробка самих моделей , сайту та додатку для iOS та Android, адміністрування серверів

та баз даних. Маркетингова кампанія за визначений час до релізу. Що стосується післяпродажного обслуговування, то це підтримка і регулярні оновлення компонентів продукту та додатку, імплементація нових функцій.

Функціональне призначення: Отримання даних та беттінг.

Підготовка користувача до користування продуктом: Значна: тісна співпраця між інженерами даних та девопсами, документація API в друкованій та електронній формі. Для використання додатку необхідно, щоб користувач мав смартфон на базі IOS або Android, вмів базово ним користуватися, потрібен доступ до інтернету, а також знання базових понять в сфері беттінгу.

Асортимент: Сайт, 2 додатки та ключі доступу до бази через API.

Технічні характеристики:

База даних:

- Тип: mongoDB.
- Доступ: API 3.0.

Додаток:

Сумісність з ОС: IOS 8.0 і старше, Android 4.4. і старше

Підтримка пристроїв: iPhone 5S +, iPad2 +, iPad Air +, iPad mini +

Верстка iPhone/iPad Книжкова: Так

Верстка iPhone/iPad Альбомна: Адаптивна від книжкової

Сервер:

Сумісний вебхостинг на базі Apache2 + PHP5 + MySQL

Основні етапи виробництва продукту:

- Збір вимог до продукту.
- Попередня оцінка вартості розробки.
- Проектування прототипу.
- Складання технічного завдання.
- Розробка першого релізу продукту.
- Тестування.
- Технічна підтримка.

Контроль якості: Постійне тестування продукту інженерами, мануальними та автоматизованими тестувальниками, знаходження баг та їх починка програмістами.

Можливості для подальшого розвитку продукту є.

Необхідні патенти та ліцензії для даного сегмента ринку:

Патент на інтелектуальну власність, ліцензії на розробку додатку і представлення його в AppStore, Play Market.

Відгуків про продукт немає, оскільки він знаходиться на етапі ідеї.

4.7.4 Аналіз ринку

Таблиця 4.22 – Таблична форма для проведення pest-аналізу.

Групи чинників	Події / чинники	- / +	Вірогідність події або прояву чинника	Важливість чинника або події	Нег / Поз	Програма дій
Політичні	1. Нестабільність законодавчої бази	-	80%	4	Нег	Має розроблятися експертами з урахуванням деталей ситуації
	2. Державна підтримка	+	20%	3	Поз	
	3. Війна	-	40%	5	Нег	
Економічні	1. Збільшення податків	-	60%	2	Нег	Має розроблятися експертами з урахуванням деталей ситуації
	2. Збільшення курсу валют	-	90%	2	Нег	
	3. Покращення платоспроможності населення	+	20%	3	Поз	
Соціальні	1. Зростання морального тиску	-	80%	2	Нег	Має розроблятися експертами з урахуванням деталей ситуації
	2. Соціальне сприйняття	+	35%	4	Поз	
Технологічні	1. Створення аналогічного продукту	-	45%	5	Нег	Має розроблятися експертами з урахуванням деталей ситуації
	2. Перехід до інших девайсів як основних для використання	+	10%	5	Поз	

Таблиця 4.23 – Матриця оцінки загальноекономічних факторів впливу зовнішнього середовища на реалізацію стартап-проекту.

Групи та окремі фактори		Оцінка залежності					Можливості	Загрози
		1	2	3	4	5		
Макроекономічні	Збільшення курсу валют				*			+
	Покращення платоспроможності населення				*		+	
	Зниження процентних ставок		*				+	
	Збільшення податкового навантаження			*				+
	Зростання економіки і розширення ринку					*	+	
Елементи конкурентного середовища	Ринкові фактори				*		+	+
	Вплив конкурентів					*		+
	Вплив споживачів					*	+	+
Соціально - демографічні	Зміни смаків споживачів, посилення рівня диференціації споживчих запитів та поява вільних ринкових сегментів, ін.				*		+	+
Зміни в системі державного регулювання	Зміни в оподаткуванні			*			+	+
	Запровадження ліцензування та обов’язкової стандартизації		*					+
	Посилення державного контролю	*						+
	Посилення корупції	*					+	+
	Регулювання ціноутворення		*				+	+
	Зміни в державних замовленнях	*					+	+
Галузеві	Посилення конкуренції					*		+
Політичні	Нестабільність законодавчої бази			*				+
	Державна політика з підтримки стартапів					*	+	
	Війна			*				+
Науково-технічні	Створення аналогічного продукту конкурентами					*		+
Природні	Перехід до інших девайсів					*	+	+

Таблиця 4.24 – SWOT-аналіз стартап-проекту.

	Сильні сторони стартапу 1. Продукт є автономним 2. Простота у використанні 3. Метод є новим, зручним та ефективним	**Слабкі сторони стартапу** 1. Продукт є новим, досі невідомим для цільового сегменту
Можливості стартапу 1.Вихід на міжнародний ринок 2.Новий функціонал	Новизна продукту допоможе вийти на міжнародний ринок. Простота у використанні дозволить додавати новий функціонал без загрози відтоку клієнтів.	Вихід на міжнародний ринок може демонстративно показати те, що продукт є дійсно якісним і корисним і прорекламувати себе для цільового сегменту.
Загрози стартапу 1. Активізація конкурентів 2. Недостатність коштів	Автономність і простота у використанні можуть виділити продукт у разі активації конкурентів.	Необхідно прорекламувати продукт і зробити його відомим для цільової аудиторії, що допоможе уникнути конкуренції і компенсувати недостатність коштів приростом нових клієнтів та партнерів.

4.8 Висновок до розділу

Розділ було присвячено опису ідеї та побудові бізнес плану для розвитку описаного стартап проекту. Розглянуто ролі учасників команд та розраховано їх долі у розробці чи підтримці продукту. Подано аналіз ринку та виокремлено переваги й напрямки стратегічного розвитку проекту.

ВИСНОВКИ

В ході написання магістерської дисертації було розглянуто рейтингові моделі на основі навчання з підкріпленням.

Першим кроком стала постановка задачі, опис елементів моделі та принципи її роботи. Розроблено схему прорахунку ймовірностей та оновлення рейтингів. Визначено більш вузьку зону дослідження, а саме – побудова моделі вибору коефіцієнту оновлення рейтингової моделі. Важливою частиною є визначення чітких критеріїв оцінювання результатів побудови моделі.

Другий та третій розділи були присвячені опису моделей вибору коефіцієнту оновлення та визначення результатів. В другому розділі ці моделі будувалися на байєсівській оптимізації , а в третьому – Q-навчанні. Зведені результати роботи цих моделей подано в наступних таблицях.

Таблиця 1 – Зведені результати за критерієм перехресної ентропії.

Модель	Перехресна ентропія
Константна модель (БО)	**0.633308**
Зі змінним коефіцієнтом оновлення (БО)	**0.630491**
Двопараметрова модель (Q-навчання)	**0.629099**
Трьопараметрова модель (Q-навчання)	**0.627954**

Таблиця 2 – Зведені результати за критерієм співпадіння розподілів на навчальних даних (2018-2019 рр.).

Модель	Розмір переваги*		
	[0.0; 0.2]	(0.2; 0.5]	(0.5; 1]
Константна модель (БО)	0.0087	0.0326	0.0471
Зі змінним коефіцієнтом оновлення (БО)	0.0147	0.0278	0.0283
Двопараметрова модель (Q-навчання)	0.0149	0.0264	0.025
Трьопараметрова модель (Q-навчання)	0.002	0.0337	0.0376

* Під розміром переваги мається на увазі модуль різниці ймовірностей для учасників.

Таблиця 3 – Зведені результати за критерієм співпадіння розподілів на ненавчальних даних (2020 р.).

Модель	Розмір переваги		
	[0.0; 0.2]	(0.2; 0.5]	(0.5; 1]
Константна модель (БО)	0.0413	0.0303	0.0500
Зі змінним коефіцієнтом оновлення (БО)	0.0408	0.0182	0.0518
Двопараметрова модель (Q-навчання)	0.0406	0.0184	0.0549
Трьопараметрова модель (Q-навчання)	0.0406	0.0186	0.0430

Також варто зазначити декілька фактів про створені моделі:

1. Медіана рейтингів є від'ємною (рисунок 1), що свідчить про те, що середньостатистичний учасник ліги є слабшим за нового.

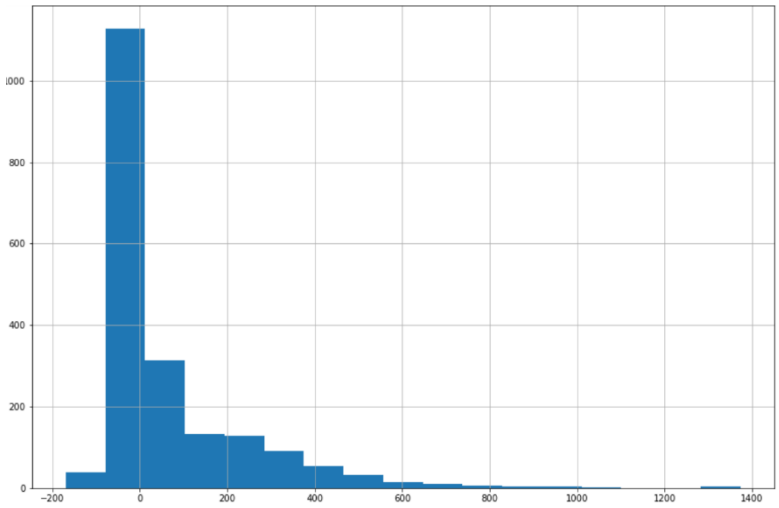

Рисунок 1 – Гістограма поточних рейтингів останньої моделі.

2. Всі члени топ 5 гравців за рейтинговою системою (рисунок 2) були або є в п'ятірці найкращих гравців, а перші двоє мають звання першої та другої ракетки світу відповідно (на час розробки моделі).

Djokovic N. (Srb)	1373.068930
Nadal R. (Esp)	1345.095730
Federer R. (Sui)	1341.795470
Soderling R. (Swe)	1062.297912
Medvedev D. (Rus)	1060.981081

Рисунок 2 – Топ 5 найкращих гравців за останньою моделлю.

Останній четвертий розділ містить в собі стартап-проект під розроблені моделі.

Дана робота має потенціал до подальшого дослідження з залученням більшої кількості даних, ускладнення моделей вибору коефіцієнту оновлення. Найперспективнішими напрямками вважаю:

1. Побудову дерева рішень з пошуком розділюючого значення на базі байєсівської оптимізації.
2. Оптимізація кількості станів Q-навчання, відсіюючи найменш ймовірні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Easterbrook, Gregg. "More flags on D spins scoreboards". ESPN. URL: https://www.espn.com/nfl/story/_/page/TMQWeekEleven141118/more-defensive-penalties-causes-change-nfl-tuesday-morning-quarterback (Last accessed: 15.12.2020)
2. Simmons, Bill. "Week 8 Picks: A Gambling Epiphany". Grantland. URL: http://grantland.com/the-triangle/week-8-picks-a-gambling-epiphany (Last accessed: 15.12.2020)
3. "The Game Designer: Pythagoras Explained". URL: http://thegamedesigner.blogspot.com/2012/05/pythagoras-explained.html (Last accessed: 15.12.2020)
4. Elo, Arpad E. "8.4 Logistic Probability as a Rating Basis". The Rating of Chessplayers, Past&Present. 2008. Bronx NY 10453: ISHI Press International. ISBN 978-0-923891-27-5.
5. Silver, Nate. "Introducing NFL Elo Ratings". FiveThirtyEight. URL: https://fivethirtyeight.com/features/introducing-nfl-elo-ratings/ (Last accessed: 15.12.2020)
6. Mills, Matt. "Using Continuous-Time Markov Chains to Rank College Football Teams". URL: http://thespread.us/continuous-markov-ratings.html (Last accessed: 15.12.2020)
7. "Ranking NFL teams using Network Science". LinkedIN. URL: https://www.linkedin.com/pulse/ranking-nfl-teams-using-network-science-konstantinos-pelechrinis?trk=prof-post/ (Last accessed: 15.12.2020)
8. Bradley, Ralph Allan; Terry, Milton E. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*. 1952. Vol. 39, No 3. P. 324–345.
9. Weng, Ruby C.; Lin, Chih-Jen. A Bayesian Approximation Method for Online Ranking (PDF). *Journal of Machine Learning Research*. 2011. No. 12. P. 267–300.

ДОДАТОК А
ВИЗНАЧЕННЯ РЕЙТИНГОВОЇ СИСТЕМИ ТА ІСНУЮЧІ ПІДХОДИ ДО РЕАЛІЗАЦІЇ РЕЙТИНГОВОЇ СИСТЕМИ

А.1 Основні поняття в сфері рейтингових систем

А.1.1 Класифікація рейтингів на основі процедури оцінювання

На рис. А.1 подано класифікацію рейтингів на основі процедури оцінювання. Кольором позначено той тип задач, що буде розглянутий в даній роботі.

Рисунок А.1 – Класифікація рейтингів на основі процедури оцінювання.

А.1.2 Визначення рейтингової системи

Для початку визначимо, що надалі називатимемо рейтинговою системою.

Рейтинговою системою $R(t)$ є набір пар елементів e_i з простору E, для якого створюється рейтингова система та чисел v_i, що є чисельним вираженням якоїсь характеристики даного елементу. Причому, чим більшим або меншим є число, тим кращим або гіршим ми вважаємо елемент.

$$\begin{cases} R = \{(e, v) \mid e \in E, v \in \mathbb{R}\}, \\ \forall\, (e_1, v_1), (e_2, v_2) \in R: (v_1 < v_2) \Rightarrow (e_1 \prec e_2) \\ \forall\, (e_1, v_1), (e_2, v_2) \in R: (v_1 = v_2) \Rightarrow (e_1 \sim e_2) \end{cases}$$

де:

$R -$ рейтингова система,

$E -$ простір рейтингових елементів.

Також, існують багатофакторні рейтингові системи, тобто ті, в яких елемент виражається набором чисел, але їх можна вважати суперпозицією декількох однофакторних рейтингових систем.

А.1.3 Визначення ігрової рейтингової системи

Ігровою рейтинговою системою є пара рейтингової системи і функції:

$$R_{game} = (R, f: \mathbb{R} \to [0, 1])$$

де f – функція двох змінних, що є апроксимацію ймовірності перемоги гравця чи команди за різницею рангів $v_1 - v_2$, тобто

$$\forall\,(e_1, v_1), (e_2, v_2) \in R: f(v_1 - v_2) = \mathbb{P}\{e_1 \text{ переможе } e_2\} + \varepsilon$$

де ε – шум.

Елементами таких рейтингових систем можуть бути як окремі особи, так і команди. Щоправда, у випадку командних рейтингових систем більш надійним підходом є визначення рейтингу для кожного гравця та їх усередненням для команди. Рахуючи командний рейтинг слід враховувати кількість попередніх ігор зіграних в подібному складі.

А.1.4 Загальна схема реалізації рейтингової моделі

На рисунку А.2 подано загальну схему реалізації, що складається з:

1. Функція оновлення рейтингової системи $f\left(r_1, r_2, \vec{\beta}\right)$, де
   - r_1 та r_2 – рейтинги команд або гравців;
   - $\vec{\beta}$ – інформація про матч, після якого відбувається оновлення.
2. База даних поточних рейтингів.

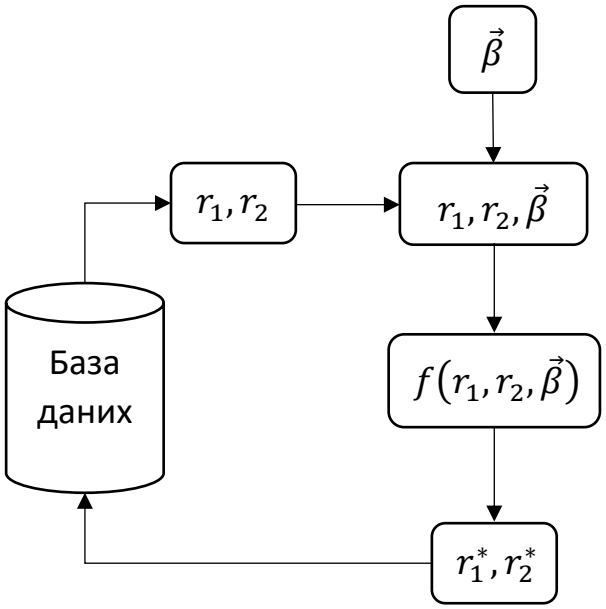

Рисунок А.2 – Схема оновлення рангів у базі даних.

А.2 Модель Ело

А.2.1 Теоретична основа моделі Ело

Загальні положення моделі:

5. Сила гри гравців в одній грі є випадковою величиною ξ.
6. Рейтинг $r = \mathbb{E}\xi$.
7. Завдання моделі – знайти найкращу оцінку $\hat{r}$ величини r.
8. Вихідною моделлю Ело є нормальний розподіл сили гри навколо рейтингу.

Маємо розподіл сили гри в заданій грі:

$$p(x) = f_{\xi \sim \mathcal{N}(r,\delta^2)}(x) = \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{(x-r)^2}{2\delta^2}}$$

де:

$r = \mathbb{E}\xi$ – рейтинг, що необхідно оцінити;

δ^2 – дисперсія, що за припущенням Ело є постійною і не залежить від гравця.

Отже, шуканими параметрами є:

$$\arg\max_{r,\delta^2} p(r, \delta^2 \mid D) = \arg\max_{r,\delta^2} \frac{p(D \mid r, \delta^2)p(r, \delta^2)}{p(D)} =$$

$$= \arg\max_{r,\delta^2} p(D \mid r, \delta^2)p(r, \delta^2)$$

де D – дані, що використовуються для навчання моделі.

Так, як Гаусів розподіл з відомою дисперсією є самоспряженим апріорним розподілом, тому якщо сила гри має нормальний розподіл навколо рейтингу, то візьмемо нормальний розподіл як апріорний для рейтингу. А отже, рейтинг

гравця або команди складається з пари (μ, σ^2), де μ – середнє значення рейтингу, а σ^2 – показник якості оцінки рейтингу.

Розглянемо гру між двома гравцями з деякими апріорними розподілами рейтингу $\mathcal{N}(r_1, \mu_1, \sigma_1^2)$ та $\mathcal{N}(r_2, \mu_2, \sigma_2^2)$. Маємо наступне твердження для сили гри кожного з гравців:

$$p(x \mid \mu, \sigma^2) = \int_{-\infty}^{+\infty} p(x \mid r)\, p(r \mid \mu, \sigma^2) dr = \int_{-\infty}^{+\infty} f_{\mathcal{N}(r,\delta^2)}(x) f_{\mathcal{N}(\mu,\sigma^2)}(r) dr =$$

$$= \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{(x-r)^2}{2\delta^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mu-r)^2}{2\sigma^2}} dr =$$

$$= \frac{1}{2\pi\delta\sigma} \int_{-\infty}^{+\infty} e^{-\left(\frac{x^2}{2\delta^2}+\frac{r^2}{2\delta^2}-\frac{2rx}{2\delta^2}+\frac{\mu^2}{2\sigma^2}+\frac{r^2}{2\sigma^2}-\frac{2r\mu}{2\sigma^2}\right)} dr =$$

$$= \frac{1}{2\pi\delta\sigma} e^{-\left(\frac{x^2}{2\delta^2}+\frac{\mu^2}{2\sigma^2}\right)} \int_{-\infty}^{+\infty} e^{-\frac{r^2(\delta^2+\sigma^2)-2r(x\sigma^2+\mu\delta^2)}{2\delta^2\sigma^2}} dr = f_{\mathcal{N}(\mu_x,\sigma_x^2)}(x)$$

Головною задачею навчання є перерахунок рейтингів після гри та врахування останніх даних.

А.2.2 Практичний приклад Ело рейтингу

За приклад Ело рейтингу візьмемо канонічну задачу оновлення рейтингів шахістів ФІДЕ(приклад співвідношення рейтингів та звань подано в таблиці А.1)

Для поточної гри гравців A і B обчислюється математичне сподівання кількості очок, які має набрати гравець:

$$\mathbb{E}_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}$$

де:

E_A – математичне сподівання кількості очок, які набере гравець A в партії з B,

R_A – попередній рейтинг гравця A,

R_b – попередній рейтинг гравця B.

Оновлений рейтинг гравця A рахується за формулою:

$$R'_A = R_A + K(S_A - \mathbb{E}_A)$$

де:

$$K = \begin{cases} 10, & R_A \geq 2400 \\ 15, & (R_A < 2400) \wedge (\text{більше 15 ігор з рейтингом ФІДЕ}) \\ 25, & \text{до 15 ігор з рейтингом ФІДЕ} \end{cases}$$

S_A – фактична кількість набраних очок гравця A,

R'_A - оновлений рейтинг гравця A.

Таблиця А.1 – Співвідношення рейтингів Ело та шахових звань і розрядів.

≥ 2500	Гросмейстер
$2400 - 2499$	Міжнародний майстер
$2200 - 2399$	Національний майстер
$2000 - 2199$	Кандидат у майстри
$1800 - 1999$	$1 -$ й розряд
$1600 - 1799$	$2 -$ й розряд
$1400 - 1599$	$3 -$ й розряд
$1200 - 1399$	Середній любитель
$1000 - 1199$	Слабкий любитель
< 1000	Новачок

А.3 Модель Бредлі–Террі

А.3.1 Алгоритм міноризуючої–максимізації

Нехай $f(\theta)$ – опукла функція, яку необхідно максимізувати, тоді на ітераційному кроці $m = 0, 1, \ldots$ функція $g(\theta|\theta_m)$ є мінімізованою функцією (сурогатною функцією) функції $f(\theta)$ в точці θ_m якщо:

$$\begin{cases} \forall\theta\colon g(\theta|\theta_m) \le f(\theta) \\ g(\theta_m|\theta_m) = f(\theta_m) \end{cases}$$

Максимізуємо сурогатну функцію $g(\theta|\theta_m)$ і визначимо:

$$\theta_{m+1} = \arg\max_{\theta} g(\theta|\theta_m)$$

Спрямувавши $m \to +\infty$, описаний ітеративний метод буде збігатися до локального оптимуму або сідлової точки, що показано на рисунку А.3

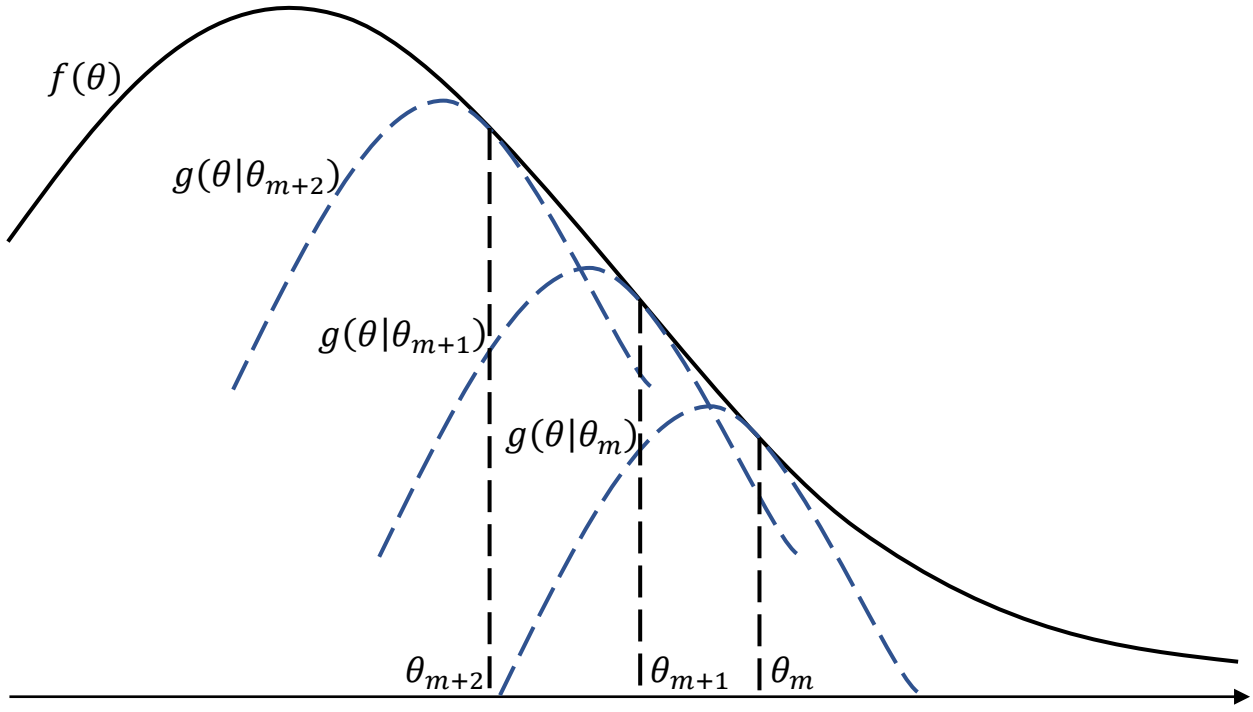

Рисунок А.3 – Візуалізація ітераційного процесу ММ – алгоритму.

Так, з попередньо вказаного алгоритму маємо наступну нерівність:

$$f(\theta_{m+1}) \geq g(\theta_{m+1}|\theta_m) \geq g(\theta_m|\theta_m) = f(\theta_m)$$

В подібний спосіб працює алгоритм Мажоризуючої–Мінімізації, для знаходження мінімуму опуклої функції.

Для побудови мажорної або мінорної версії цільової функції можна застосовувати будь-які нерівності.

А.3.2 Теоретична основа моделі Бредлі–Террі

Основна ідея:

Для гравців $g = \{g_1, \dots, g_n\}$ можливо підібрати такі рейтинги $y = \{y_1, \dots, y_n\}$, що:

$$\mathbb{P}\{y_i \succ y_j\} = \frac{y_i}{y_i + y_j}$$

Основна задача:

Знайти $y = \{y_1, \dots, y_m\}$ максимальної правдоподібності з наявних даних D. Визначивши апріорний розподіл як рівномірний, маємо:

$$p(D \mid y) = \prod_{i=1}^{m} \prod_{j=1}^{m} \left(\frac{y_i}{y_i + y_j}\right)^{w_{ij}} \to max$$

де w_{ij} – кількість перемог g_i над g_j $(w_{ii} = 0)$.

Це теж саме, що:

$$I(y) = \ln p(D \mid y) = \sum_{i=1}^{m}\sum_{j=1}^{m}\left(w_{ij}\ln y_i - w_{ij}\ln\left(y_i + y_j\right)\right) \to max$$

Максимізацію функції $I(y)$ будемо проводити за допомогою алгоритму Міноризуючої–Максимізації. Для початку створимо сурогатну функцію. Розглянемо функцію $f(x) = 1 + \ln x - x$ на проміжку $(0;\ +\infty)$ (рисунок А.4):

$$\begin{cases} \lim\limits_{x\to 0+0} f(x) = -\infty \\ \lim\limits_{x\to +\infty} f(x) = -\infty \\ f'(x) = \dfrac{1}{x} - 1 \\ f''(x) = -\dfrac{1}{x^2} < 0 \end{cases} \Rightarrow \exists!\, x = 1 - \text{т. максимуму} \Rightarrow f(x) \le f(1) \Rightarrow$$

$$\Rightarrow 1 + \ln x - x \le 0$$

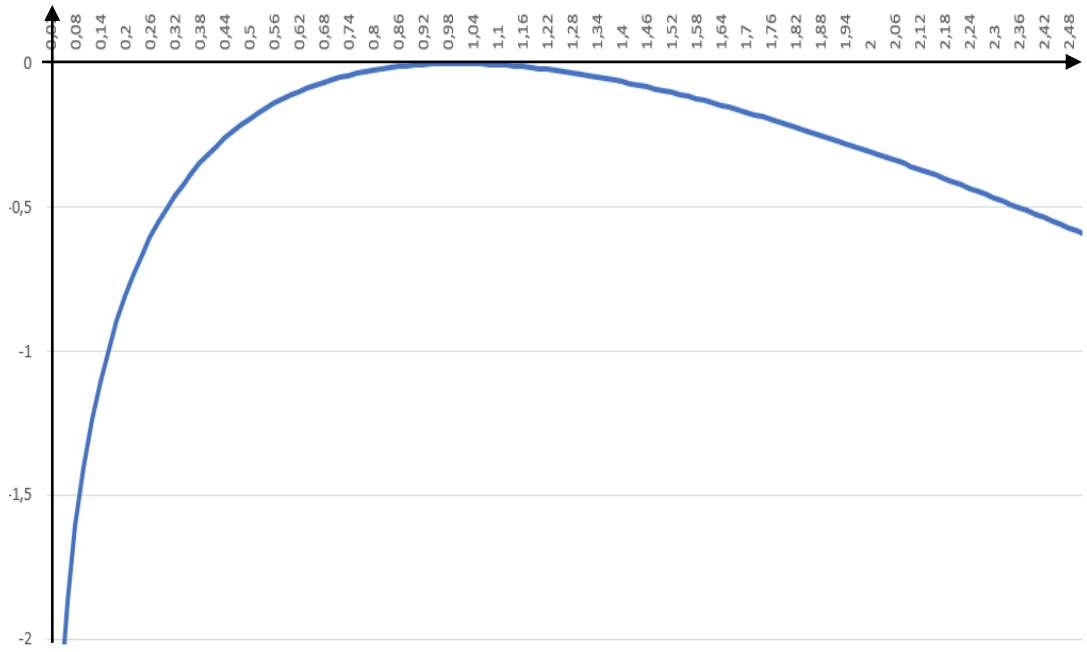

Рисунок А.4 – Візуалізація функції $f(x) = 1 + \ln x - x$.

На основі отриманої нерівності маємо наступну сурогатну функцію $Q(y, y^m)$ функції $I(y)$:

$$I(y) = \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\ln y_i - \ln\left(y_i + y_j\right)\right) \geq$$

$$\geq \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\ln y_i - \ln\left(y_i + y_j\right) + 1 + \ln\frac{y_i + y_j}{y_i^k + y_j^k} - \frac{y_i + y_j}{y_i^k + y_j^k}\right) =$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\ln y_i - \ln\left(y_i + y_j\right) + 1 + \ln\left(y_i + y_j\right) - \ln\left(y_i^k + y_j^k\right) - \frac{y_i + y_j}{y_i^k + y_j^k}\right) =$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\ln y_i - \ln\left(y_i^k + y_j^k\right) - \frac{y_i + y_j}{y_i^k + y_j^k} + 1\right) = Q(y, y^k) \Rightarrow$$

$$\Rightarrow I(y) \geq Q(y, y^k)$$

Візьмемо похідну від $Q(y, y^m)$ по y_I:

$$\frac{\partial Q}{\partial y_I}(y, y^k) = \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\frac{\delta_{iI}}{y_i} - \frac{\delta_{iI} + \delta_{jI}}{y_i^k + y_j^k}\right) - \sum_{j=1}^{m}\frac{w_{Ij}}{y_i^k + y_j^k} - \sum_{i=1}^{m}\frac{w_{iI}}{y_i^k + y_j^k}$$

Так, як:

1. w_I – загальна кількість перемог гравця $I \Rightarrow w_I = \sum_{j}^{m} w_{Ij}$.
2. N_{ij} – кількість зустрічей між гравцями i та $j \Rightarrow N_{ij} = w_{ij} + w_{ji}$.

$$\frac{w_I}{y_I} - \sum_{j=1}^{m}\frac{N_{Ij}}{y_i^k + y_j^k} = 0$$

В результаті маємо наступну функцію оновлення рангів:

$$y_I^k := w_I \left(\sum_{j=1}^{m} \frac{N_{Ij}}{y_i^k + y_j^k} \right)^{-1}$$

А.3.3 Приклади модифікацій моделі

1. Врахування нічиї:

$$\exists\, \theta > 1: \begin{cases} \mathbb{P}\{i \succ j\} = \dfrac{y_i}{y_i + \theta y_j}, \\ \mathbb{P}\{j \succ i\} = \dfrac{y_j}{y_j + \theta y_i}, \\ \mathbb{P}\{i \sim j\} = \dfrac{(\theta^2 - 1) y_i y_j}{(y_i + \theta y_j)(y_i + \theta y_j)} \end{cases}$$

2. Домашні і виїзні матчі:

$$\exists\, \theta: \mathbb{P}\{i \succ j\} = \begin{cases} \dfrac{\theta y_i}{\theta y_i + y_j}, \text{якщо } i \text{ грає вдома} \\ \dfrac{y_j}{\theta y_i + y_j}, \text{якщо } i \text{ грає на виїзді} \end{cases}$$

де $\theta -$ коефіцієнт переваги домашньої команди.

3. Турнірні матчі:

$$\mathbb{P}_A(\pi) = \prod_{i=1}^{m} \frac{y_{\pi(i)}}{\sum_{k=i}^{m} y_{\pi(k)}}, \pi - \text{перестановка підмножини } A = \{1, \dots, m\}$$

А.4 Рейтингова система TrueSkill

А.4.1 Алгоритм розповсюдження довіри

Алгоритм розповсюдження довіри – алгоритм маргіналізації (знаходження часткового розподілу) за допомогою двонаправленої передачі повідомлень по графу. Застосовується для виводу на графічних ймовірнісних моделях, таких як байєсові та маркові мережі.

Розглянемо функцію:

$$p^*(X) = \prod_{j=1}^{m} f_j(X_j), \qquad \text{де } X_j = \{x_i\}_{i=1}^{n}$$

Задача маргіналізації:

$$\text{Знайти } p_i^*(x_i) = \sum_{k \neq i} p^*(X)$$

Задача нормалізованої маргіналізації:

$$\text{Знайти } p_i(x_i) = \sum_{k \neq i} p(X), \text{де } p(X) = \frac{1}{Z}\prod_{j=1}^{m} f_j(X_j), Z = \sum_{X}\prod_{j=1}^{m} f_j(X_j)$$

Всі ці задачі, в найгіршому випадку мають експоненціальне зростання складності, але в деяких випадках їх можливо розв’язати швидше, для чого й використовується алгоритм.

Граф, що використовується алгоритмом, складається з вершин-змінних та вершин-функцій, причому функції поєднанні з тими змінними, від яких вони залежать.

Для прикладу візьмемо функцію

$$p^*(X) = f_1(x_1)f_2(x_2, x_3)f_3(x_1, x_3)f_4(x_3)f_5(x_2, x_3)$$

Тоді відповідний граф має вигляд, що зображено на рисунку А.5.

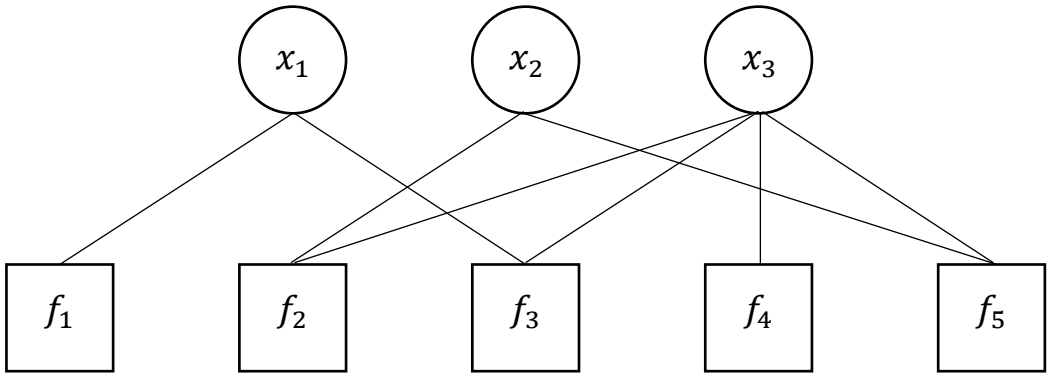

Рисунок А.5 – Граф залежностей функцій від змінних для функції $p^*(X)$.

По графу відсилаються повідомлення двох видів:

1. Від змінної до функції

$$q_{i\to j}(x_i) = \prod_{k \in ne(i) Jay} r_{k\to i}(x_i)$$

де $ne(i) -$ множина вершин, сусідніх з i

2. Від функції до змінної

$$r_{j\to i}(x_i) = \sum_{X_i \setminus x_i} f_j(X_j) \prod_{k \in ne(i)\setminus j} r_{k\to j}(x_k)$$

Пустий добуток вважаємо рівним одиниці.

Зазначимо, що якщо у вершини всього один сусід, то її повідомлення можливо вирахувати не знаючи вхідного повідомлення. Функція f_j, що залежить

лише від однієї змінної x_i передає своїй змінній $r_{j\to i} = f_j(x_i)$, а змінна x_i, від якої залежить лише одна функція, передаватиме їй 1.

За типом графу маємо 2 варіанти алгоритму:

1. Граф є деревом:

   Починаючи з листків будемо проходити всі вершини і вираховувати повідомлення, враховуючи правило: повідомлення можливо надіслати, якщо його можливо повністю побудувати. Такий алгоритм має кінцеву кількість кроків, що дорівнює діаметру графу, тобто довжині найдовшого шляху.

2. Граф не є деревом:

   Для початку вважатимемо, що всі змінні передають повідомлення 1, а в подальшому будемо модифікувати їх, коли до них доходитимуть повідомлення від функцій. В загальному випадку такий алгоритм робить багато непотрібних кроків і не завжди працює коректно.

Після закінчення роботи алгоритму, маргінали розраховуються за формулою:

$$p_i^*(x_i) = \prod_{j \in ne(i)} r_{j\to i}(x_i)$$

$$p_i(x)_i = \frac{1}{Z} p_i^*(x_i), \text{де } Z = \sum_i p_i^*(x_i)$$

Якщо необхідно розрахувати нормалізовані маргінали, тобто справжні ймовірності, то потрібно нормалізувати повідомлення від змінних до функцій:

$$q_{i\to j}(x_i) = \alpha_{ij} \prod_{k \in ne(i)Jay} r_{k\to i}(x_i)$$

де α_{ij} підібрані так, щоб

$$\sum_i q_{i\to j}(x_i) = 1$$

Ідеєю алгоритму є перескладання початкового добутку

$$p^*(X) = \prod_{j=1}^{m} f_j(X_j)$$

в

$$p^*(X) = \prod_{j=1}^{m} \phi_j(X_j) \prod_{\text{ш}=1}^{m} \psi_i(x_i)$$

де ϕ_j відповідає вершинам-функціям, а ψ_i – вершинам-змінним.

Початковими значеннями є:

$$\begin{cases} \phi_j(X_j) = f_j(X_j) \\ \psi_i(x_i) = 1 \end{cases}$$

В подальшому, при отриманні повідомлення від функції до змінної, ϕ та ψ перераховуються за формулами:

$$\psi_i(x_i) = \prod_{j \in ne(i)} r_{j\to i}(x_i)$$

$$\phi_j(X_j) = \frac{f_j(X_j)}{\prod_{i \in ne(j)} r_{j\to i}(x_i)}$$

В процесі добуток є незмінним, а кінцеві ψ_i після передачі всіх повідомлень стануть маргіналами $p_i^*(x_i)$.

А.4.2 Опис рейтингової системи TrueSkill

Основною задачею рейтингової системи TrueSkill є розрахунок апостеріорних рейтингів гравців, які об’єднуються в команди різного розміру, після кожного з таких турнірів.

Так як ми не маємо дійсних апріорних значень рейтингів, але ми маємо тільки деякий апріорний розподіл, який ми вважаємо нормальним

$$f(s_i) = \mathcal{N}(s;\ \mu_i, \sigma_i)$$

де μ_i – власне рейтинг гравця

σ_i – показник достовірності рейтингу, тобто його дисперсія.

Кожний дійсний рейтинг є середнім значенням, навколо якого розподілені конкретні показники сили гри того чи іншого гравця в конкретній грі або партії

$$f(p_i \mid s_i) = \mathcal{N}(p_i;\ s_i, \beta^2)$$

Як і в моделі Ело, будемо вважати β^2 – універсальна константа.

Тоді p_i через вихідні параметри виражається як:

$$f(p_i \mid \mu_i, \sigma_i) = \int_{-\infty}^{+\infty} \mathcal{N}(p_i;\ s_i, \beta^2)\mathcal{N}(s_i;\ \mu_i, \sigma_i)ds_i$$

Показники сили гри гравців в конкретних іграх або партіях об’єднуються і дають оцінки сили гри команд. В системі TrueSkill припускається, що сила команди дорівнює сумі сил її гравців:

$$t_i = \sum_i p_i.$$

Після цього показники сил команд у турнірі необхідно порівняти один з одним, при чому їх порівняння повинно створювати той же порядок, що записаний в результатах турніру.

Будемо вважати, що нічия між командами з силою t_i та t_j означає, що:

$$\exists \varepsilon = const: \forall i, j: \left|t_i - t_j\right| < \varepsilon$$

Після отримання даних, необхідно підрахувати апостеріорні рейтинги команд.

Дані ми отримуємо, як впорядковані результати турніру, які можуть містити нічия між сусідніми командами. Позначимо ці впорядковані результати як перестановку команд π.

Отже, нам необхідно підрахувати наступний вираз:

$$\mathbb{P}\{\boldsymbol{s} \mid \pi\} = \frac{\mathbb{P}\{\pi \mid \boldsymbol{s}\}\mathbb{P}\{\boldsymbol{s}\}}{\int \mathbb{P}\{\pi \mid \boldsymbol{s}\}\mathbb{P}\{\boldsymbol{s}\} d\boldsymbol{s}}$$

В нашій системі, окрім s_i та π, існують змінні p_i, t_i та d_i. Щільність всієї системи являє собою добуток розподілів:

$$\mathbb{P}\{\pi, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{p}, \boldsymbol{s}\} = \mathbb{P}\{\pi \mid \boldsymbol{d}\}\mathbb{P}\{\boldsymbol{d} \mid \boldsymbol{t}\}\mathbb{P}\{\boldsymbol{t} \mid \boldsymbol{p}\}\mathbb{P}\{\boldsymbol{p} \mid \boldsymbol{s}\}\mathbb{P}\{\boldsymbol{s}\}$$

А отже ми маємо порахувати наступний інтеграл:

$$\mathbb{P}\{\pi \mid \boldsymbol{s}\} = \iiint \mathbb{P}\{\pi, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{p}, \boldsymbol{s}\} \cdot d\boldsymbol{d} \cdot d\boldsymbol{t} \cdot d\boldsymbol{s}$$

Отримуємо звичайну задачу маргіналізації, що розв’язується алгоритмом розповсюдженням довіри.

На рисунках А.6, А.7 та А.8 подано приклади дерев для різних типів турнірів.

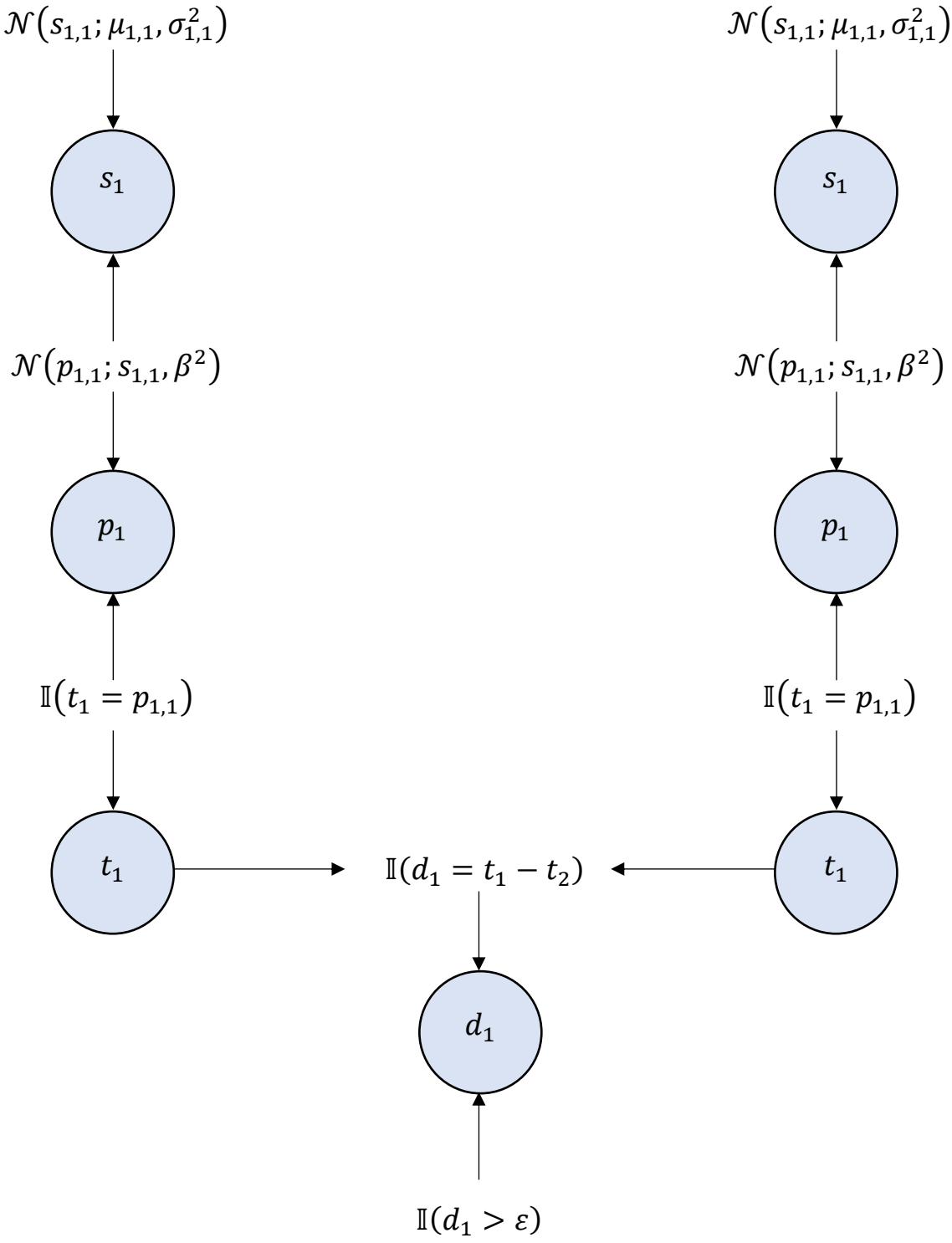

Рисунок А.6 – Приклад розрахункового графу для двох гравців.

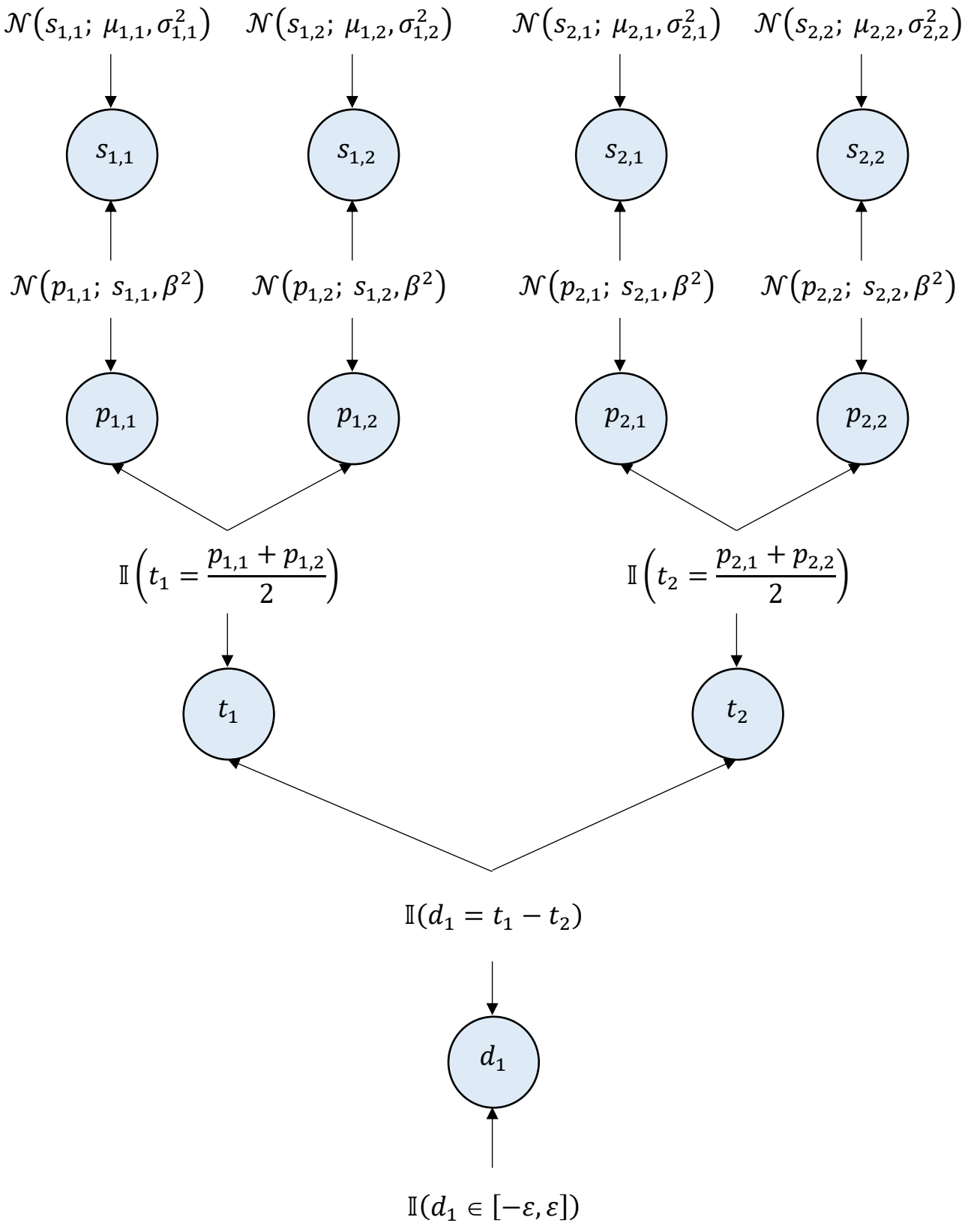

Рисунок А.7 – Приклад розрахункового графу для двох команд, що складаються з двох гравців.

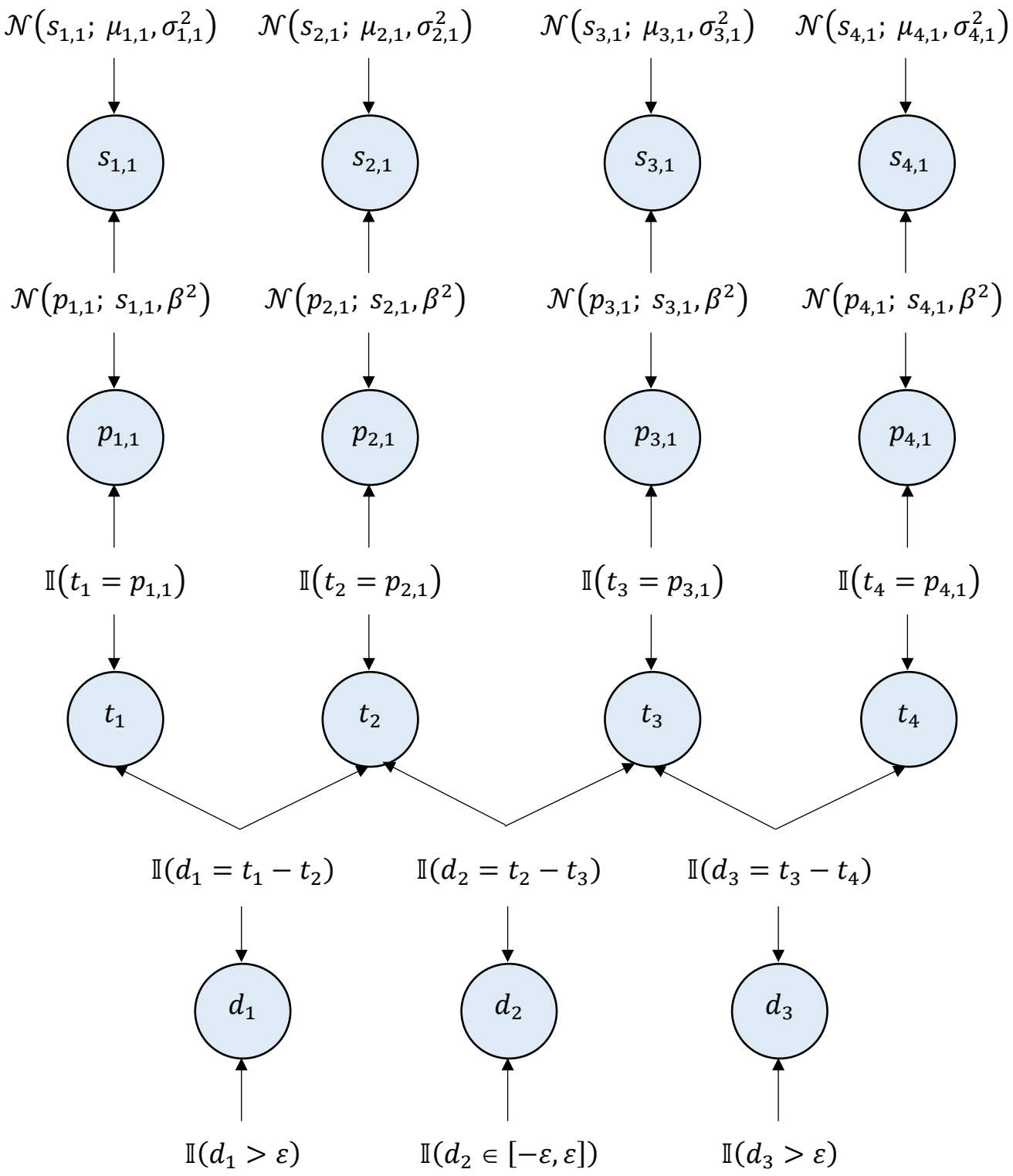

Рисунок А.8 – Приклад розрахункового графу турніру з чотирьох гравців

Виникає питання: який вигляд має функція на найнижчому рівні, тобто та яка визначає перемогу чи нічию? (рисунок А.9)

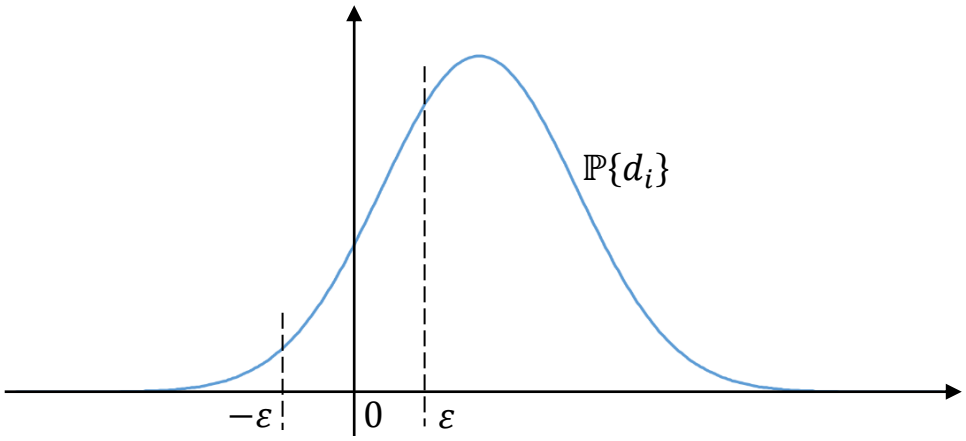

Рисунок А.9 – Залежність ймовірності перемоги від d_i.

У випадку перемоги маємо (рисунок А.10):

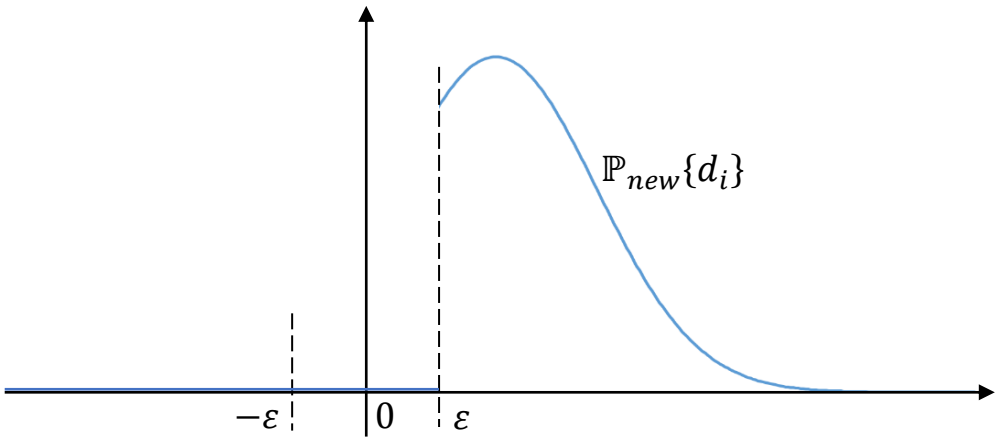

Рисунок А.10 – Вигляд кінцевої функції у разі перемоги.

Для випадку нічиї (рисунок А.11):

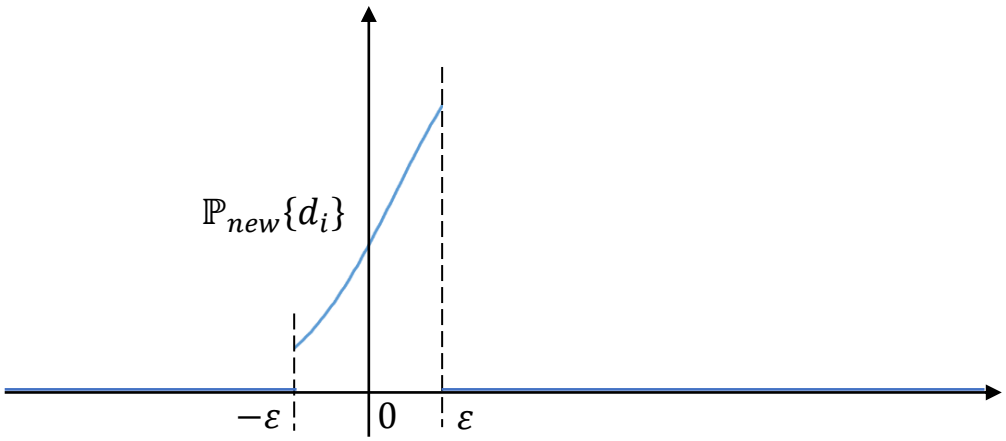

Рисунок А.11 – Вигляд кінцевої функції у разі нічиї.

Для отримання результатів з нижнього рівня використовується алгоритм Expectation Propagation:

1. Наближуємо повідомлення від функції до змінної (розподіл) деяким сімейством розподілів.
2. Передаємо повідомлення взад-вперед до співпадіння оцінок.

В даному випадку будемо використовувати нормальні розподіли, тож для отримання оптимальних наближень потрібно знайти перші два моменти розподілу: мат сподівання та дисперсію.

ДОДАТОК Б
ЛІСТИНГ ПРОГРАМИ

Download.py:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:
import re
import time

import numpy as np
import pandas as pd

from bs4 import BeautifulSoup
from selenium import webdriver
from urllib.request import urlopen
from datetime import datetime, timedelta

ROOT_PATH = "https://www.scoreboard.com"
SINGE_MATCHES_FILENAME = "./../../data/dataframes/single_matches.csv"

# pd.DataFrame(columns=[
#     'tournament', 'season', 'date', 'time', 'home_player', 'away_player', 'home_score', 'away_score',
'home_part_scores', 'away_part_scores'
# ]).to_csv(SINGE_MATCHES_FILENAME, index=False)

single_matches = pd.read_csv(SINGE_MATCHES_FILENAME)
if single_matches.shape[0] > 0:
    single_matches_last_season = single_matches.tail(1).iloc[-1]["season"]
    single_matches_last_tournament = single_matches.tail(1).iloc[-1]["tournament"]
else:
    single_matches_last_season = ""
    single_matches_last_tournament = ""

soup = BeautifulSoup(urlopen(ROOT_PATH + "/en/tennis/"), 'html.parser')

atp_single = soup.find("li", attrs={"id": "lmenu_5724"})
atp_double = soup.find("li", attrs={"id": "lmenu_5726"})

atp_single_tournaments = [tournament for tournament in atp_single.find_all("li") if
single_matches_last_tournament == "" or tournament.text >= single_matches_last_tournament]
atp_double_tournaments = atp_double.find_all("li")

single_matches_last_tournament, single_matches_last_season
```

```
with webdriver.Chrome("C:/Users/Vlado4ka/Downloads/chromedriver_win32/chromedriver.exe")
as driver:
    for tournament_source in atp_single_tournaments:
        tournament_results_url = ROOT_PATH + tournament_source.find("a").attrs["href"] +
"archive/"
        tournament_name = tournament_source.text
        driver.get(tournament_results_url)
        time.sleep(2)
        seasons = BeautifulSoup(driver.page_source).find_all("div", attrs={"class":
"leagueTable__season"})
        if single_matches_last_tournament == tournament_name:
            seasons = [season for season in seasons if single_matches_last_season == "" or season.text
<= single_matches_last_season]
        for season in seasons:
            if "leagueTable__season--heading" not in season.attrs["class"]:
                season_name = season.text
                season_url = ROOT_PATH + season.find("a").attrs["href"] + "results/"
                print(season_url)
                driver.get(season_url)
                time.sleep(2)
                soup = BeautifulSoup(driver.page_source)
                year = int(soup.find("div", attrs={"id": "fscountry"}).find("div", attrs={"class":
"teamHeader__information"}).find("div", attrs={"class": "teamHeader__text"}).text)

                matches = soup.find_all("div", attrs={"title": "Click for match detail!"})
                final_month = int(matches[0].find("div", attrs={"class": "event__time"}).text[3:5])

                for match in matches:
                    datetime_source = match.find('div', attrs={'class': 'event__time'})
                    while datetime_source.find("div") is not None:
                        datetime_source.find("div").extract()
                        datetime_source = match.find('div', attrs={'class': 'event__time'})
                        print(".")
                    if re.match(r"\d{2}\.\d{2}\. \d{2}:\d{2}", datetime_source.text):
                        record = {
                            "tournament": tournament_name,
                            "season": season_name,
                            "date": datetime.strptime(f"{datetime_source.text.split(' ')[0]}{year}",
"%d.%m.%Y").date(),
                            "time": datetime.strptime(f"{datetime_source.text.split(' ')[1]}",
"%H:%M").time(),
                            "home_player": match.find('div', attrs={'class': 'event__participant--home'}).text,
                            "away_player": match.find('div', attrs={'class': 'event__participant--away'}).text,
                            "home_score": match.find('div', attrs={'class': 'event__score--home'}).text,
                            "away_score": match.find('div', attrs={'class': 'event__score--away'}).text,
                            "home_part_scores":  [elem.text for elem in match.find_all('div', attrs={'class':
'event__part--home'})],
                            "away_part_scores":  [elem.text for elem in match.find_all('div', attrs={'class':
'event__part--away'})],

                        }
                        if record["date"].month > final_month:
```

```
                record["date"] = record["date"].replace(year=record["date"].year - 1)

pd.DataFrame([pd.Series(record)]).to_csv(SINGE_MATCHES_FILENAME, index=False,
header=False, mode="a")
print("Success!")

sm = pd.read_csv("./../../data/dataframes/single_matches.csv").drop_duplicates()
sm.to_csv("./../../data/dataframes/single_matches.csv", index=False)
sm.to_csv("./../../data/duplicates/single_matches.csv", index=False)
```

Preprosess.py:

```
#!/usr/bin/env python
# coding: utf-8

import re

import numpy as np
import pandas as pd

sm = pd.read_csv("./../../data/duplicates/single_matches.csv")

sm["winner_home"] = np.where(sm["home_score"] > sm["away_score"], 1, 0)
sm["winner_home"] = np.where(sm["home_score"] == sm["away_score"], 0.5,
sm["winner_home"])

sm = sm.drop(["home_part_scores", "away_part_scores"], axis=1)
sm = sm.sort_values(["date", "time"]).reset_index(drop=True)

sm.head(3)

def get_country(string: str):
    searched = re.search(r"(?!\()\w+(?=\))", string)
    if searched:
        return searched.group()
    return "None"

def get_clear_name(string: str):
    matched = re.match(r"^[^\s]{2,}(\s|-).+\([A-Z][a-z]+\)$", string)
    if matched:
        return string.split("(")[0].strip()
    return "None"


players = pd.DataFrame(pd.concat([sm["home_player"], sm["away_player"]]), columns=["label"])
players["country"] = players["label"].map(get_country)
players["name"] = players["label"].map(get_clear_name)
redundant_players = players.loc[players["country"] == "None", "label"].unique()
players = players[(players["country"] != "None") & (players["name"] != "None")]
players = players.drop_duplicates()

players.head(3)
```

```
sm = sm[~(sm["home_player"].isin(redundant_players) |
sm["away_player"].isin(redundant_players))]
sm.to_csv("./../../data/dataframes/single_matches_timeline.csv", index=False)

ranks = players[["label"]].copy()
ranks["rank"] = 0
ranks["games_count"] = 0

ranks.to_csv("./../../data/dataframes/ranks.csv", index=False)
```

BO1feature.py:

```
import featexp
import plotly.graph_objects as go

import numpy as np
import pandas as pd

ranks = pd.read_csv("./../../data/dataframes/ranks.csv")
data_df = pd.read_csv("./../../data/dataframes/single_matches_timeline.csv", parse_dates=["date"])

data_df = data_df[data_df["date"] >= "2008-01-01"]


def get_player_rank(players, player_name):
    return players.get(player_name, 0)


def get_player_games_count(counts, player_name):
    return counts.get(player_name, 0)


def predict_home_win_chance(home_rank, away_rank, scalar):
    return 1.0 / (1.0 + np.exp((away_rank - home_rank) / scalar))


def calc_new_rank(old_rank, win_prob, win_res, k):
    return old_rank + k * (win_res - win_prob)


def log_loss(real, pred):
    if (len(real) == len(pred)) & (min(min(real), min(pred)) >= 0.0) & (max(max(real), max(pred)) <= 1.0):
        return - sum([r * np.log(p) + (1 - r) * np.log(1 - p) for r, p in zip(real, pred)]) / len(real)
    return None


def rank_model(data, scaler, k, players, counts):
    real = []
    predicted = []
    home_ranks = []
    away_ranks = []

    for row in data.itertuples(index=False):
        home_rank = get_player_rank(players, row.home_player)
```

```
        away_rank = get_player_rank(players, row.away_player)

        home_ranks.append(home_rank)
        away_ranks.append(away_rank)

        home_games_count = get_player_games_count(counts, row.home_player)
        away_games_count = get_player_games_count(counts, row.away_player)

        home_win_prob = predict_home_win_chance(home_rank, away_rank, scalar=scaler)

        new_home_rank = calc_new_rank(home_rank, home_win_prob, row.winner_home, k)

        new_away_rank = calc_new_rank(away_rank, 1.0 - home_win_prob, 1.0 - row.winner_home, k)

        players.update({row.home_player: new_home_rank, row.away_player: new_away_rank})
        counts.update({row.home_player: home_games_count + 1, row.away_player: away_games_count + 1})

        real.append(row.winner_home)
        predicted.append(home_win_prob)

    data["home_rank"] = home_ranks
    data["away_rank"] = away_ranks
    data["win_chance"] = predicted

    return data


def f(X):
    k = X[:, 0]
    res = rank_model(data_df.copy(deep=True), 300, k,
                     ranks[["label", "rank"]].set_index("label").to_dict()["rank"],
                     ranks[["label", "games_count"]].set_index("label").to_dict()["games_count"])
    test = res[res["date"].map(lambda x: x.year).isin([2018, 2019])]
    return log_loss(test["winner_home"], test["win_chance"])


domain = [
    {"name": "k", "type": "discrete", "domain": np.arange(40.0, 50.0, 0.1)},
]

from GPyOpt.methods import BayesianOptimization

optimizer = BayesianOptimization(f=f,
                                 domain=domain,
                                 num_cores=4,
                                 initial_design_numdata=3,
                                 acquisition_type="LCB",
                                 model_type="GP")

optimizer.run_optimization(max_iter=10, eps=1e-8)

optimizer.plot_acquisition()

optimizer.plot_convergence()

print("Optimized Parameters:\n",
      domain[0]["name"], optimizer.x_opt[0], "\n",
```

```
    )
print("optimized loss: ", optimizer.fx_opt)

player_ranks = ranks[["label", "rank"]].set_index("label").to_dict()["rank"]
player_games = ranks[["label", "games_count"]].set_index("label").to_dict()["games_count"]

ranked = rank_model(data_df, 300, 43.9,
                    player_ranks,
                    player_games)

final_player_ranks = pd.Series(player_ranks)
final_player_ranks = final_player_ranks[final_player_ranks != 0.0]

final_player_ranks.hist(bins=17, figsize=(15, 10))

final_player_ranks.median()

pd.DataFrame(final_player_ranks.sort_values(ascending=False).head(10), columns=["rank"])

ranked["year"] = ranked["date"].map(lambda x: x.year)

ranked201819 = ranked[ranked["year"].isin([2018, 2019])]
ranked2020 = ranked[ranked["year"] == 2020]


def simple_test(df, bins=10):
    df = df[["winner_home", "win_chance"]]
    doubled = df.copy()
    doubled = 1 - doubled
    df = pd.concat([df, doubled])

    df = featexp.get_grouped_data(df, "win_chance", "winner_home", bins)[1]

    fig = go.Figure([
        go.Scatter(x=df["win_chance_mean"], y=df["winner_home_mean"], mode="markers"),
        go.Scatter(x=[0.0, 1.0], y=[0.0, 1.0], mode="lines")
    ])
    fig.update_layout(autosize=False, width=600, height=600, )
    fig.show()
    return df


a = simple_test(ranked201819)

c = a[((a["win_chance_mean"] > 0.0) & (a["win_chance_mean"] < 0.25)) |
      ((a["win_chance_mean"] > 0.75) & (a["win_chance_mean"] < 1.0))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()

a = simple_test(ranked2020)

c = a[((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6)) |
      ((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()

a["winner_home_mean"].map(lambda x: round(x, 3))
```

BO5features.py:

```
import featexp
import plotly.graph_objects as go

import numpy as np
import pandas as pd

from GPyOpt.methods import BayesianOptimization

ranks = pd.read_csv("./../../data/dataframes/ranks.csv")
data_df = pd.read_csv("./../../data/dataframes/single_matches_timeline.csv", parse_dates=["date"])

data_df = data_df[data_df["date"] >= "2008-01-01"]


def simple_test(df, bins=10):
    df = df[["winner_home", "win_chance"]]
    doubled = df.copy()
    doubled = 1 - doubled
    df = pd.concat([df, doubled])

    df = featexp.get_grouped_data(df, "win_chance", "winner_home", bins)[1]

    fig = go.Figure([
        go.Scatter(x=df["win_chance_mean"], y=df["winner_home_mean"], mode="markers"),
        go.Scatter(x=[0.0, 1.0], y=[0.0, 1.0], mode="lines")
    ])
    fig.update_layout(autosize=False, width=600, height=600, )
    fig.show()
    return df


def get_player_rank(players, player_name):
    return players.get(player_name, 0)


def get_player_games_count(counts, player_name):
    return counts.get(player_name, 0)


def predict_home_win_chance(home_rank, away_rank, scalar):
    return 1.0 / (1.0 + np.exp((away_rank - home_rank) / scalar))


def calc_new_rank(old_rank,
                  games_count,
                  win_prob, win_res,
                  k_min_time, min_time,
                  k_best_players, best_players_threshold, k_standard):
    if games_count <= 89.0:
        k = 63.0
    elif games_count <= min_time:
```

```
        k = k_min_time
    elif old_rank > 740:
        k = 77.
    elif old_rank > best_players_threshold:
        k = k_best_players
    else:
        k = k_standard
    return old_rank + k * (win_res - win_prob)


def log_loss(real, pred):
    if (len(real) == len(pred)) & (min(min(real), min(pred)) >= 0.0) & (max(max(real), max(pred))
<= 1.0):
        return - sum([r * np.log(p) + (1 - r) * np.log(1 - p) for r, p in zip(real, pred)]) / len(real)
    return None


def rank_model(data, scalar, k_min_time, min_time, k_best_players, best_players_threshold,
k_standard, players, counts):
    predicted = []
    home_ranks = []
    away_ranks = []

    for row in data.itertuples(index=False):
        home_rank = get_player_rank(players, row.home_player)
        away_rank = get_player_rank(players, row.away_player)

        home_games_count = get_player_games_count(counts, row.home_player)
        away_games_count = get_player_games_count(counts, row.away_player)

        home_win_prob = predict_home_win_chance(home_rank, away_rank, scalar)

        new_home_rank = calc_new_rank(home_rank,
                                      home_games_count,
                                      home_win_prob,
                                      row.winner_home,
                                      k_min_time, min_time,
                                      k_best_players, best_players_threshold, k_standard)

        new_away_rank = calc_new_rank(away_rank,
                                      away_games_count,
                                      1.0 - home_win_prob,
                                      1.0 - row.winner_home,
                                      k_min_time, min_time,
                                      k_best_players, best_players_threshold, k_standard)

        players.update({row.home_player: new_home_rank, row.away_player: new_away_rank})
        counts.update({row.home_player: home_games_count + 1, row.away_player:
away_games_count + 1})

        predicted.append(home_win_prob)
        home_ranks.append(home_rank)
```

```
        away_ranks.append(away_rank)

    data["home_rank"] = home_ranks
    data["away_rank"] = away_ranks
    data["win_chance"] = predicted
    return data


def f(X):
    k_min_time = X[:, 0]
    min_time = X[:, 1]
    k_best_players = X[:, 2]
    best_players_threshold = X[:, 3]
    k_standard = X[:, 4]

    res = rank_model(data_df,
                     300,
                     k_min_time,
                     min_time,
                     k_best_players,
                     best_players_threshold,
                     k_standard,
                     ranks[["label", "rank"]].set_index("label").to_dict()["rank"],
                     ranks[["label", "games_count"]].set_index("label").to_dict()["games_count"]
                     )
    test = res[res["date"].map(lambda x: x.year).isin([2018, 2019])]
    return log_loss(test["winner_home"], test["win_chance"])


domain = [
    {"name": "k_min_time", "type": "discrete", "domain": tuple(i for i in range(20, 66, 2))},
    {"name": "min_time", "type": "discrete", "domain": tuple(i for i in range(250, 321, 2))},
    {"name": "k_best_players", "type": "discrete", "domain": tuple(i for i in range(20, 60, 2))},
    {"name": "best_players_threshold", "type": "discrete", "domain": tuple(i for i in range(400, 561,
10))},
    {"name": "k_standard", "type": "discrete", "domain": tuple(i for i in range(20, 40, 2))},
]

optimizer = BayesianOptimization(f=f, domain=domain, num_cores=4, initial_design_numdata=7,
acquisition_type="LCB")

optimizer.run_optimization(max_iter=100)

print("Optimized Parameters:\n",
      domain[0]["name"], optimizer.x_opt[0], "\n",
      domain[1]["name"], optimizer.x_opt[1], "\n",
      domain[2]["name"], optimizer.x_opt[2], "\n",
      domain[3]["name"], optimizer.x_opt[3], "\n",
      domain[4]["name"], optimizer.x_opt[4], "\n",
      "\n")
print("optimized loss: ", optimizer.fx_opt)
```

```
player_ranks = ranks[["label", "rank"]].set_index("label").to_dict()["rank"]
player_games = ranks[["label", "games_count"]].set_index("label").to_dict()["games_count"]

ranked = rank_model(data_df, 300,
                    64, 90,
                    76, 750, 30, player_ranks, player_games)

final_player_ranks = pd.Series(player_ranks)
final_player_ranks = final_player_ranks[final_player_ranks != 0.0]
final_player_ranks.hist(bins=15, figsize=(15, 10))

final_player_ranks.median()

ranked["year"] = ranked["date"].map(lambda x: x.year)

ranked201819 = ranked[ranked["year"].isin([2018, 2019])]
ranked2020 = ranked[ranked["year"] == 2020]

a = simple_test(ranked201819)

c = a[((a["win_chance_mean"] > 0.25) & (a["win_chance_mean"] < 0.4)) |
     ((a["win_chance_mean"] > 0.6) & (a["win_chance_mean"] < 0.75))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()

a = simple_test(ranked2020)

c = a[((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6)) |
     ((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()
```

Model1q.py:

```
import featexp
import plotly.graph_objects as go

import numpy as np
import pandas as pd

ranks_df = pd.read_csv("./../../data/dataframes/ranks.csv")
data_df = pd.read_csv("./../../data/dataframes/single_matches_timeline.csv", parse_dates=["date"])

data_df = data_df[data_df["date"] >= "2008-01-01"]

steps = np.array([20, 63, 77, 29, 38, 30])
ranks = np.array([0, 400, 740])
counts = np.array([89, 300])


class Training:
   def __init__(self, lr, discount):
```

```
        self.random = 15
        self.discount = discount
        self.train = True
        self.lr = lr

    def close(self):
        self.train = False


training = Training(1e-4, 0.99)


def get_index(value, arr):
    if value <= arr[0]:
        return 0
    return np.where(arr < value)[0][-1] + 1


def get_q_row_index(rank, count):
    return get_index(rank, ranks) * (len(counts) + 1) + get_index(count, counts)


def simple_test(df, bins=10):
    df = df[["winner_home", "win_chance"]]
    doubled = df.copy()
    doubled = 1 - doubled
    df = pd.concat([df, doubled])

    df = featexp.get_grouped_data(df, "win_chance", "winner_home", bins)[1]

    fig = go.Figure([
        go.Scatter(x=df["win_chance_mean"], y=df["winner_home_mean"], mode="markers"),
        go.Scatter(x=[0.0, 1.0], y=[0.0, 1.0], mode="lines")
    ])
    fig.update_layout(autosize=False, width=600, height=600, )
    fig.show()
    return df


def predict_home_win_chance(home_rank, away_rank, scalar):
    return 1.0 / (1.0 + np.exp((away_rank - home_rank) / scalar))


def log_loss(real, pred):
    if (len(real) == len(pred)) & (min(min(real), min(pred)) >= 0.0) & (max(max(real), max(pred))
<= 1.0):
        return - sum([r * np.log(p) + (1 - r) * np.log(1 - p) for r, p in zip(real, pred)]) / len(real)
    return None


def calc_new_rank(old_rank, win_prob, win_res, Q, Q_updates, training, game_count, player_id,
states):
```

```
    state = get_q_row_index(old_rank, game_count)
    action = np.argmax(Q[state])
    k = steps[action]
    res = old_rank + k * (win_res - win_prob)

    if training.train:
        best_next_state = np.max(Q[get_q_row_index(res, game_count + 1)])
        Q_updates[state][action] += training.lr * (training.discount * best_next_state -
Q[state][action])
        if states.get(player_id) is not None:
            prev_state, prev_act = states[player_id]
            reward = win_res * np.log(win_prob) + (1.0 - win_res) * np.log(1.0 - win_prob)
            Q_updates[prev_state][prev_act] += training.lr * reward
        states.update({player_id: (state, action)})
    return res, Q_updates


def log_loss(real, pred):
    if (len(real) == len(pred)) & (min(min(real), min(pred)) >= 0.0) & (max(max(real), max(pred))
<= 1.0):
        return - sum([r * np.log(p) + (1 - r) * np.log(1 - p) for r, p in zip(real, pred)]) / len(real)
    return None


def rank_model(data, scalar, Q, players, game_counts, states):
    predicted = []
    home_ranks = []
    away_ranks = []

    Q_updates = np.zeros_like(Q)

    if training.train:
        mask = np.repeat(np.random.choice([0.0, 1.0], size=(Q.shape[0], 1), p=[0.7, 0.3]),
Q.shape[1]).reshape(Q.shape)
        Q_with_random = Q * (1 - mask) + mask * np.random.uniform(0, 1, size=mask.shape)
    else:
        Q_with_random = Q

    for row in data.itertuples(index=False):
        home_rank = players.get(row.home_player, 0.0)
        home_count = game_counts.get(row.home_player, 0.0)

        away_rank = players.get(row.away_player, 0.0)
        away_count = game_counts.get(row.away_player, 0.0)

        home_win_prob = predict_home_win_chance(home_rank, away_rank, scalar)

        lr = training.lr
        if row.date.year not in [2018, 2019]:
            training.lr = 0.0

        new_home_rank, Q_updates = calc_new_rank(home_rank,
```

```
                                    home_win_prob,
                                    row.winner_home,
                                    Q_with_random, Q_updates, training, home_count,
row.home_player,
                                    states)

        new_away_rank, Q_updates = calc_new_rank(away_rank,
                                    1.0 - home_win_prob,
                                    1.0 - row.winner_home,
                                    Q_with_random, Q_updates, training, away_count,
row.away_player,
                                    states)

        training.lr = lr

        players.update({row.home_player: new_home_rank, row.away_player: new_away_rank})
        game_counts.update({row.home_player: home_count + 1, row.away_player: away_count +
1})

        predicted.append(home_win_prob)
        home_ranks.append(home_rank)
        away_ranks.append(away_rank)

    data["home_rank"] = home_ranks
    data["away_rank"] = away_ranks
    data["win_chance"] = predicted

    #     if training.train & (row.date >= "2019-01-01"):
    #         training.lr *= 0.99

    return data, Q_updates, np.float64(Q_with_random == Q_with_random.max(axis=1,
keepdims=True))


Q_old = Q_new = np.random.uniform(-0.01, 0.01, size=((len(ranks) + 1) * (len(counts) + 1),
len(steps)))
min_ = 1.

for _ in range(500):
    player_ranks = ranks_df[["label", "rank"]].set_index("label").to_dict()["rank"]
    players_states = {}
    data, Q_upd, Q_mask = rank_model(data_df, 300, Q_old, player_ranks, {}, players_states)

    test = data[data["date"].map(lambda x: x.year).isin([2018, 2019])]
    loss = log_loss(test["winner_home"], test["win_chance"])

    if loss - min_ < 1e-8:
        bestQ = Q_old
        min_ = loss
        Q_upd += 10 * Q_mask * (1.0 - loss)

    Q_new = Q_old + Q_upd
```

```
    Q_old = Q_new
    print(loss)

training.train = False

player_ranks = ranks_df[["label", "rank"]].set_index("label").to_dict()["rank"]
players_states = {}
ranked, _, _ = rank_model(data_df, 300, bestQ, player_ranks, {}, players_states)

final_player_ranks = pd.Series(player_ranks)
final_player_ranks = final_player_ranks[final_player_ranks != 0.0]
final_player_ranks.hist(bins=17, figsize=(15, 10))

ranked["year"] = ranked["date"].map(lambda x: x.year)

ranked201819 = ranked[ranked["year"].isin([2018, 2019])]
ranked2020 = ranked[ranked["year"] == 2020]

a = simple_test(ranked201819)

c = a[((a["win_chance_mean"] > 0.0) & (a["win_chance_mean"] < 0.25)) |
     ((a["win_chance_mean"] > 0.75) & (a["win_chance_mean"] < 1.0))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()

a = simple_test(ranked2020)

c = a[((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6)) |
     ((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()
```

Model2q.py:

```
import re
import featexp
import plotly.graph_objects as go

import numpy as np
import pandas as pd

ranks_df = pd.read_csv("./../../data/dataframes/ranks.csv")
data_df = pd.read_csv("./../../data/dataframes/single_matches_timeline.csv", parse_dates=["date"])

data_df = data_df[data_df["date"] >= "2008-01-01"]
data_df = data_df[data_df["home_score"].map(lambda x: True if re.match(r"\d", x) else False)]
data_df["delta"] = data_df["home_score"].astype(float) - data_df["away_score"].astype(int)

steps = np.array([29, 32, 35, 38, 41, 45, 61, 64, 67, 70])
ranks = np.array([0, 400, 740])
counts = np.array([89, 300, 600])
deltas = np.array([-2, -1, 1, 2])
```

```
class Training:
    def __init__(self, lr, discount):
        self.random = 15
        self.discount = discount
        self.train = True
        self.lr = lr

    def close(self):
        self.train = False


training = Training(1e-4, 0.99)


def get_index(value, arr):
    if value <= arr[0]:
        return 0
    return np.where(arr < value)[0][-1] + 1


def get_q_row_index(rank, count, delta):
    return get_index(rank, ranks) * (len(counts) + 1) * (len(deltas) + 1) + get_index(count, counts) *
(
            len(deltas) + 1) + get_index(delta, deltas)


def simple_test(df, bins=10):
    df = df[["winner_home", "win_chance"]]
    doubled = df.copy()
    doubled = 1 - doubled
    df = pd.concat([df, doubled])

    df = featexp.get_grouped_data(df, "win_chance", "winner_home", bins)[1]

    fig = go.Figure([
        go.Scatter(x=df["win_chance_mean"], y=df["winner_home_mean"], mode="markers"),
        go.Scatter(x=[0.0, 1.0], y=[0.0, 1.0], mode="lines")
    ])
    fig.update_layout(autosize=False, width=600, height=600, )
    fig.show()
    return df


def predict_home_win_chance(home_rank, away_rank, scalar):
    return 1.0 / (1.0 + np.exp((away_rank - home_rank) / scalar))


def log_loss(real, pred):
    if (len(real) == len(pred)) & (min(min(real), min(pred)) >= 0.0) & (max(max(real), max(pred))
<= 1.0):
```

```
        return - sum([r * np.log(p) + (1 - r) * np.log(1 - p) for r, p in zip(real, pred)]) / len(real)
    return None


def calc_new_rank(old_rank, win_prob, win_res, Q, Q_updates, training, game_count, delta,
player_id, states):
    state = get_q_row_index(old_rank, game_count, delta)
    action = np.argmax(Q[state])
    k = steps[action]
    res = old_rank + k * (win_res - win_prob)

    if training.train:
        best_next_state = np.max(
            Q[get_q_row_index(res, game_count + 1, min(deltas)): get_q_row_index(res, game_count +
1, max(deltas) + 1)])
        Q_updates[state][action] += training.lr * (training.discount * best_next_state -
Q[state][action])
        if states.get(player_id) is not None:
            prev_state, prev_act = states[player_id]
            reward = win_res * np.log(win_prob) + (1.0 - win_res) * np.log(1.0 - win_prob)
            Q_updates[prev_state][prev_act] += training.lr * reward
        states.update({player_id: (state, action)})
    return res, Q_updates


def log_loss(real, pred):
    if (len(real) == len(pred)) & (min(min(real), min(pred)) >= 0.0) & (max(max(real), max(pred))
<= 1.0):
        return - sum([r * np.log(p) + (1 - r) * np.log(1 - p) for r, p in zip(real, pred)]) / len(real)
    return None


def rank_model(data, scalar, Q, players, game_counts, states):
    predicted = []
    home_ranks = []
    away_ranks = []

    Q_updates = np.zeros_like(Q)

    if training.train:
        mask = np.repeat(np.random.choice([0.0, 1.0], size=(Q.shape[0], 1), p=[0.7, 0.3]),
Q.shape[1]).reshape(Q.shape)
        Q_with_random = Q * (1 - mask) + mask * np.random.uniform(0, 1, size=mask.shape)
    else:
        Q_with_random = Q

    for row in data.itertuples(index=False):
        home_rank = players.get(row.home_player, 0.0)
        home_count = game_counts.get(row.home_player, 0.0)

        away_rank = players.get(row.away_player, 0.0)
        away_count = game_counts.get(row.away_player, 0.0)
```

```
        home_win_prob = predict_home_win_chance(home_rank, away_rank, scalar)

        lr = training.lr
        if row.date.year not in [2018, 2019]:
            training.lr = 0.0

        new_home_rank, Q_updates = calc_new_rank(home_rank,
                                                 home_win_prob,
                                                 row.winner_home,
                                                 Q_with_random, Q_updates, training, home_count, row.delta,
                                                 row.home_player, states)

        new_away_rank, Q_updates = calc_new_rank(away_rank,
                                                 1.0 - home_win_prob,
                                                 1.0 - row.winner_home,
                                                 Q_with_random, Q_updates, training, away_count, -row.delta,
                                                 row.away_player, states)

        training.lr = lr

        players.update({row.home_player: new_home_rank, row.away_player: new_away_rank})
        game_counts.update({row.home_player: home_count + 1, row.away_player: away_count +
1})

        predicted.append(home_win_prob)
        home_ranks.append(home_rank)
        away_ranks.append(away_rank)

    data["home_rank"] = home_ranks
    data["away_rank"] = away_ranks
    data["win_chance"] = predicted
    return data, Q_updates, np.float64(Q_with_random == Q_with_random.max(axis=1,
keepdims=True))


Q_old = Q_new = np.random.uniform(-0.01, 0.01,
                          size=((len(ranks) + 1) * (len(counts) + 1) * (len(deltas) + 1), len(steps)))
min_ = 1.

for _ in range(500):
    player_ranks = ranks_df[["label", "rank"]].set_index("label").to_dict()["rank"]
    players_states = {}
    data, Q_upd, Q_mask = rank_model(data_df, 300, Q_old, player_ranks, {}, players_states)

    test = data[data["date"].map(lambda x: x.year).isin([2018, 2019])]
    loss = log_loss(test["winner_home"], test["win_chance"])

    if loss - min_ < 1e-8:
        bestQ = Q_old
        min_ = loss
        Q_upd += Q_mask * (1.0 - loss)
```

```
    Q_new = Q_old + Q_upd
    Q_old = Q_new
    print(loss)

training.train = False

player_ranks = ranks_df[["label", "rank"]].set_index("label").to_dict()["rank"]
players_states = {}
ranked, _, _ = rank_model(data_df, 300, bestQ, player_ranks, {}, players_states)

final_player_ranks = pd.Series(player_ranks)
final_player_ranks = final_player_ranks[final_player_ranks != 0.0]
final_player_ranks.hist(bins=17, figsize=(15, 10))

pd.DataFrame(final_player_ranks.sort_values(ascending=False).iloc[:5], columns=["rank"])

ranked["year"] = ranked["date"].map(lambda x: x.year)

ranked201819 = ranked[ranked["year"].isin([2018, 2019])]
ranked2020 = ranked[ranked["year"] == 2020]

a = simple_test(ranked201819)

c = a[((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6)) |
     ((a["win_chance_mean"] > 0.4) & (a["win_chance_mean"] < 0.6))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()

a = simple_test(ranked2020)

c = a[((a["win_chance_mean"] > 0.0) & (a["win_chance_mean"] < 0.25)) |
     ((a["win_chance_mean"] > 0.75) & (a["win_chance_mean"] < 1.0))]

np.abs(c["win_chance_mean"] - c["winner_home_mean"]).mean()
```